

**Redes neurais de segmentação semântica  
de plantas daninhas usando mosaico  
de imagens de alta resolução espacial  
de um veículo aéreo não tripulado**

Lilian Nogueira de Faria

TESE APRESENTADA AO  
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA  
DA UNIVERSIDADE DE SÃO PAULO  
PARA OBTENÇÃO DO TÍTULO  
DE DOUTORA EM CIÊNCIAS

Programa: Ciência da Computação  
Orientador: Prof. Dr. Roberto Hirata Junior

Durante parte do desenvolvimento deste trabalho, a autora recebeu auxílio financeiro do CNPq.

São Paulo

Novembro de 2023





**Redes neurais de segmentação semântica  
de plantas daninhas usando mosaico  
de imagens de alta resolução espacial  
de um veículo aéreo não tripulado**

Lilian Nogueira de Faria

Esta versão da Tese contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 21/11/2023. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

Prof. Dr. Roberto Hirata Junior (orientador) / IME-USP

Prof. Dr. Roberto Marcondes Cesar Junior / IME-USP

Dr. Thiago Teixeira Santos / EMBRAPA

Prof. Dr. José Alberto Quintanilha / USP

Prof. Dr. André Ricardo Backes / UFSCar

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

*“Pobre de ti se pensas ser vencido!  
Tua derrota é caso decidido.  
Queres vencer, mas como em ti não crês,  
tua descrença esmaga-te de vez.*

*Se imaginas perder, perdido estás.  
Quem não confia em si, marcha pra trás;  
a força que te impele para frente  
é a decisão firmada em tua mente.”*

*(Amado Nervo)*



# Agradecimentos

Gostaria de agradecer a todos que contribuíram de alguma forma para minha formação acadêmica e realização deste trabalho de pesquisa.

Primeiramente, agradeço a Deus e ao apoio de minha família, principalmente nos momentos mais difíceis e desafiadores ao longo deste percurso.

Ao meu orientador, Prof. Dr. Roberto Hirata Junior, por me apresentar esta interessante área de pesquisa científica sobre redes neurais de aprendizado profundo em aplicações de visão computacional. Em especial, agradeço pela orientação e dicas importantes para revisão desta tese, sempre com muito respeito durante meu processo de aprendizado.

Agradeço aos professores doutores do IME-USP, João Eduardo Ferreira, Marcelo Gomes de Queiroz, Nina Sumiko Tomita Hirata, Paulo André Vechiatto de Miranda, Roberto Hirata Junior e Roberto Marcondes Cesar Junior, pelos ensinamentos nas disciplinas da Pós-Graduação em Ciência da Computação. Aos funcionários do Departamento de Computação do IME, em especial, à secretária da pós-graduação, Kátia Kiesshau, pela ajuda inestimável em alguns momentos – vocês fazem toda a diferença na nossa vida acadêmica.

Agradeço aos membros da banca de exame de qualificação, Prof. Dr. Roberto Marcondes (IME-USP) e Dr. Thiago Teixeira Santos (Embrapa), pelas sugestões para melhoria deste trabalho. Aos membros da comissão julgadora de defesa desta tese de doutorado, agradeço por aceitarem participar e avaliar o trabalho, bem como pelas valiosas contribuições e sugestões para aprimorar o texto.

Ao Laboratório Nacional de Ciência e Tecnologia do Bioetanol (CTBE), Campinas-SP, pelas imagens aéreas de drones de um campo de cana-de-açúcar. Agradeço também à equipe da empresa PIX4D, pela licença de teste do software PIX4Dmapper para geração dos ortomosaicos de alta resolução, imprescindíveis para execução deste trabalho.

Agradeço aos administradores do Laboratório de Visão Computacional do IME-USP que colaboraram para manter os recursos computacionais sempre disponíveis. Em especial, aos colegas de doutorado, Caio Rodrigues e Artur Oliveira, pelo auxílio em alguns momentos para execução prática deste trabalho.

Ao reitor da USP, Carlos Gilberto Carlotti Júnior, pela gestão humanizada durante a pandemia, bem como a todos da universidade que colaboraram direta ou indiretamente para realização deste doutorado – muito obrigada!

Para finalizar, agradeço ao auxílio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) durante os quatro primeiros anos do doutorado, sem o qual não seria possível me dedicar integralmente à pesquisa e desenvolvimento deste projeto.



# Resumo

Lilian Nogueira de Faria. **Redes neurais de segmentação semântica de plantas daninhas usando mosaico de imagens de alta resolução espacial de um veículo aéreo não tripulado.** Tese (Doutorado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Uma tecnologia moderna de sensoriamento remoto para mapeamento aéreo usando veículos aéreos não tripulados (VANTs), popularmente conhecidos como drones, tem conquistado cada vez mais espaço na agricultura de precisão devido à sua capacidade de obter ortomosaicos georreferenciados de campos agrícolas com alta resolução espacial, permitindo a localização e manejo de plantas daninhas durante todas as fases de desenvolvimento da cultura. Apesar dos importantes avanços nos sistemas de aquisição de VANTs e do desenvolvimento de novas técnicas de aprendizado de máquina usando redes neurais convolucionais (CNNs) em tarefas de classificação de imagens, a detecção automática de ervas daninhas continua sendo um problema desafiador devido à sua forte similaridade espectral com a cultura agrícola, uma vez que elas refletem o mesmo comprimento de onda eletromagnético captado pelos sensores. Com a disponibilidade de sensores RGB e multiespectrais que captam imagens de alta resolução espacial nos comprimentos de onda visível e infravermelho, técnicas mais modernas de aprendizado profundo (*deep learning*) vêm sendo aplicadas nos últimos anos para abordar o problema de segmentação semântica em aplicações de agricultura de precisão. Assim, este trabalho inclui uma revisão bibliográfica de alguns modelos de redes neurais de classificação de imagens e segmentação semântica, bem como uma análise de desempenho quantitativo e qualitativo dos mapas de previsão de nove modelos em quatro classes semânticas (solo, cultura, daninhas e gramíneas), aplicados em um domínio específico com conjuntos de dados formados por imagens e mosaicos RGB de alta resolução de um campo experimental de cultivo de cana-de-açúcar com presença de plantas daninhas. Todas as abordagens de redes de segmentação semântica baseadas em CNN diminuem a resolução espacial dos mapas de atributos de alto nível no topo da CNN e utilizam diferentes estratégias para recuperar a informação espacial para previsão semântica com a mesma dimensão da imagem de entrada. A análise dos resultados indicou que as redes que utilizam estratégias para agregar informações de contexto multiescala para segmentação de objetos de diferentes tamanhos obtêm melhores desempenhos de IoU médio em imagens com GSDs (*Ground Sampling Distance*) diferentes do utilizado no treinamento. Como os mosaicos de imagens de VANT apresentam variações de resolução espacial devido a variações de altura durante o voo, um novo modelo de segmentação semântica multiescala baseado em CNNs foi desenvolvido neste trabalho, levando em consideração as características dos modelos de melhor desempenho neste domínio específico, obtendo assim um desempenho relativamente superior ao das redes multiescala já existentes. Consequentemente, os mapas ortomosaicos georreferenciados, segmentados pela rede proposta, podem ser usados em sistemas de agricultura de precisão para localização das plantas daninhas em talhões de uma cultura agrícola, permitindo a pulverização automatizada de herbicidas nos locais certos nas doses necessárias, reduzindo os impactos ambientais e aumentando a produtividade agrícola.

**Palavras-chave:** visão computacional, inteligência artificial, aprendizado de máquina, rede neural convolucional, sensoriamento remoto, ortomosaico, agricultura de precisão.





# Abstract

Lilian Nogueira de Faria. **Neural networks for semantic segmentation of weeds using high spatial resolution image mosaic from an unmanned aerial vehicle.** Thesis (Doctorate). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

A modern remote sensing technology for aerial mapping using unmanned aerial vehicles (UAVs), popularly known as drones, has conquered more and more space in precision agriculture due to its ability to obtain georeferenced orthomosaics of agricultural fields with high spatial resolution, allowing the detection and management of weeds during all stages of crop development. Despite important advances in UAV acquisition systems and the development of new machine learning techniques using convolutional neural networks (CNNs) in image classification tasks, automatic weed detection remains a challenging problem due to their strong spectral similarity with the agricultural crop, since they reflect the same electromagnetic wavelength captured by the sensors. With the availability of RGB and multispectral sensors that capture high spatial resolution imagery in visible and infrared wavelengths, modern deep learning techniques have been applied in recent years to address the semantic segmentation problem in precision agriculture applications. Therefore, this work includes a bibliographical review of some neural network models for image classification and semantic segmentation, as well as a qualitative and quantitative performance analysis of the prediction maps of nine models in four semantic classes (soil, crop, weeds and grasses), applied in a specific domain with datasets formed by images and high-resolution RGB mosaics of an experimental sugarcane crop field with the presence of weeds. All semantic segmentation network approaches based on CNN decrease the spatial resolution of the high-level feature maps on top of the CNN and use different strategies to recover the spatial information for semantic prediction with the same dimension as the input image. The performances of these networks were compared quantitatively and qualitatively on our dataset. Analysis of the results indicated that networks that use strategies to aggregate multiscale context information to segment objects of different sizes obtain better average IoU performance in images with GSDs (Ground Sampling Distance) different from that used in training. As UAV image mosaics present variations in spatial resolution due to height variations during flight, a new multiscale semantic segmentation model based on CNNs was developed in this work, considering the characteristics of the best performing models in this specific domain, thus achieving a relatively higher performance than existing multiscale networks. Consequently, georeferenced orthomosaic maps, segmented by the proposed network, can be used in precision agriculture systems to locate weeds in fields of an agricultural crop, allowing the automated spraying of herbicides in the right locations at the necessary doses, reducing environmental impacts and increasing agricultural productivity.

**Keywords:** computer vision, artificial intelligence, machine learning, deep learning, convolutional neural network, remote sensing, orthomosaic, precision agriculture.



# Lista de abreviaturas

ANAC	Agência Nacional de Aviação Civil
AP	Agricultura de Precisão
ARM	<i>Attention Refinement Module</i>
ASPP	<i>Atrous Spatial Pyramid Pooling</i>
AUC	<i>Area Under Curve</i>
BiSeNet	<i>Bilateral Segmentation Network</i>
BN	<i>Batch Normalization</i>
CamVid	<i>Cambridge-driving Labeled Video Database</i>
CCD	<i>Charge Coupled Device</i>
CIR	<i>Color Infrared</i>
CMOS	<i>Complementary Metal Oxide Semiconductor</i>
CNN	<i>Convolutional Neural Network</i>
COCO-Stuff	<i>Common Objects in COntext-Stuff dataset</i>
CP	Centro de Projeção
CRF	<i>Conditional Random Field</i>
CTBE	Laboratório Nacional de Ciência e Tecnologia do Bioetanol
DA	<i>Data augmentation</i>
DL	<i>Deep Learning</i>
D[C]NN	<i>Deep [Convolutional] Neural Network</i>
DSM	<i>Digital Surface Model</i>
DTM	<i>Digital Terrain Model</i>
DT	<i>Decision Trees</i>
ExG	<i>Excess Green Vegetation Index</i>
EXIF	<i>Exchangeable Image File Format</i>
FC	<i>Fully Connected</i>
FCN	<i>Fully Convolutional Network</i>
FEAGRI	Faculdade de Engenharia Agrícola - Unicamp
FFM	<i>Feature Feature Module</i>
FLOPS	<i>FLoating-point Operations Per Second</i>
FoV	<i>Field of View</i>
FRRN	<i>Full Resolution Residual Network</i>
FT	<i>Fine Tuning</i>
GAP	<i>Global Average Pooling</i>
GCP	<i>Ground Control Point</i>
GeoTIFF	<i>Georeferenced TIFF</i>
GIS	<i>Geographic Information System</i>
GLI	<i>Green Leaf Index</i>
GNDVI	<i>Green Normalized Difference Vegetation Index</i>
GNSS-RTK	<i>Global Navigation Satellite System – Real Time Kinematic</i>
GPS	<i>Global Positioning System</i>
GPU	<i>Graphics Processing Unit</i>
GSD	<i>Ground Sample Distance</i>
IA	Inteligência Artificial
ILSVRC	<i>ImageNet Large Scale Visual Recognition Challenge</i>
IMU	<i>Inertial Measurement Unit</i>

IoU	<i>Intersection over Union</i>
ISO	<i>International Standards Organization</i>
JPEG	<i>Joint Photographic Experts Group</i>
KITTI	<i>Karlsruhe Institute of Technology and Toyota Technological Institute</i>
KML	<i>Keyhole Markup Language</i>
KNN	<i>K-Nearest Neighbors</i>
LCI	<i>Leaf Chlorophyll Index</i>
LRR	<i>Laplacian Pyramid Reconstruction and Refinement</i>
LZW	<i>Lempel-Ziv-Welch</i>
MCARI	<i>Modified Chlorophyll Absorption in Reflective Index</i>
MDE	<i>Modelo Digital de Elevação</i>
MDS	<i>Modelo Digital de Superfície</i>
MDT	<i>Modelo Digital de Terreno</i>
mIoU	<i>Mean Intersection over Union</i>
ML	<i>Machine Learning</i>
MLP	<i>MultiLayer Perceptron</i>
MNIST	<i>Modified National Institute of Standards and Technology</i>
MRFN	<i>Multi Receptive Field Network</i>
MsC	<i>Multi-Scale</i>
MS-COCO	<i>Microsoft Common Objects in COntext</i>
MSE	<i>Mean Squared Error</i>
NDRE	<i>Normalized Difference Red Edge Index</i>
NDVI	<i>Normalized Difference Vegetation Index</i>
NGRDI	<i>Normalized Green-Red Difference Index</i>
NIR	<i>Near Infrared</i>
PAN	<i>Pyramid Attention Network</i>
PASCAL-VOC	<i>PASCAL Visual Object Classes</i>
PNG	<i>Portable Network Graphic</i>
PSPNet	<i>Pyramid Scene Parsing Network</i>
QGIS	<i>Quantum GIS</i>
RAM	<i>Random Access Memory</i>
ReLU	<i>Rectified Linear Unit</i>
ResNet	<i>Residual Network</i>
RF	<i>Random Forest</i>
RGB	<i>Red-Green-Blue</i>
RNA	<i>Redes Neurais Artificiais</i>
SDS	<i>Simultaneous Detection and Segmentation</i>
SfM	<i>Structure from Motion</i>
SIFT	<i>Scale Invariant Feature Transform</i>
SIG	<i>Sistemas de Informação Geográfica</i>
SIPI2	<i>Structure Intensive Pigment Index 2</i>
SGD	<i>Stochastic Gradient Descent</i>
SLIC	<i>Simple Linear Iterative Clustering</i>
SPP	<i>Spatial Pyramid Pooling</i>
SRC	<i>Sistema de Referência de Coordenadas</i>
SVM	<i>Support Vector Machine</i>
<i>tanh</i>	<i>Tangente hiperbólica</i>
TGI	<i>Triangular Greenness Index</i>
TIFF	<i>Tagged Image File Format</i>
UTM	<i>Universal Transverse Mercator</i>
VANT	<i>Veículo aéreo não tripulado</i>
VARI	<i>Visible Atmospherically Resistant Index</i>
VGG	<i>Visual Geometry Group</i>
WGS84	<i>World Geodetic System 1984</i>
WRN	<i>Wide Residual Network</i>

# Lista de figuras

- Figura 1.1** – *Sensoriamento remoto com diferentes áreas de cobertura do solo (a), dependendo da altura da plataforma (satélite, aeronave ou veículo aéreo não tripulado); imagens sobrepostas ao longo de uma trajetória de voo sobre a região de interesse (b).* ..... 2
- Figura 1.2** – *Estereoscopia<sup>2</sup> para reconstruir um objeto 3D (terreno) imageado por pelo menos dois pontos de vista diferentes (imagens estereoscópicas da cena) [JENSEN 2009].*..... 3
- Figura 2.1** – *Inteligência artificial, aprendizado de máquina e aprendizado profundo.* ... 9
- Figura 2.2** – *Treinamento e teste de um modelo de aprendizado de máquina.* ..... 10
- Figura 2.3** – *Tarefas de visão computacional: classificação dos objetos; classificação e localização de caixa delimitadora de um objeto; detecção de vários objetos; e segmentação semântica densa a nível de pixel. Adaptado de [LI et al. 2017a].* ..... 11
- Figura 2.4** – *Arquitetura de uma rede neural multicamada totalmente conectada, composta por uma camada de entrada, duas camadas ocultas e uma camada de saída [LECUN 1989].*..... 12
- Figura 2.5** – *Representação de um neurônio artificial. Um vetor de entrada  $x$  é linearmente combinado com um vetor de pesos  $w$ , passando por uma função de ativação  $\theta$  para gerar a ativação de saída  $a_j$  de um neurônio  $j$  da camada  $l$ .*..... 13
- Figura 2.6** – *Funções de ativação saturante sigmóides e não saturante ReLU [NAIR e HINTON 2010].* ..... 14
- Figura 2.7** – *Exemplo simplificado de propagação direta para cálculo da saída de uma camada totalmente conectada, usando multiplicações sucessivas de matriz-vetor.* ..... 14
- Figura 2.8** – *Gap entre erro de treinamento e teste de acordo com a complexidade do modelo de rede e quantidade de dados de treinamento. Adaptado de [ABU-MOSTAFA et al. 2012].* ..... 18
- Figura 2.9** – *Influência da complexidade do modelo no ajuste aos dados de treinamento e capacidade de generalização (underfitting e overfitting). Adaptado de [ZHANG et al. 2020].*..... 19
- Figura 2.10** – *Aprendizado de máquina tradicional: extrator de atributos e classificador tradicional treinável (a); Rede neural convolucional de aprendizado profundo treinada de ponta a ponta (b). Adaptado de [LECUN et al. 1998].* ..... 20

<b>Figura 2.11</b> – <i>Arquitetura LeNet-5 com 7 camadas [LECUN et al. 1998; ZHANG et al. 2020].</i>	21
<b>Figura 2.12</b> – <i>Mapas de atributos da rede LeNet-5.</i>	22
<b>Figura 2.13</b> – <i>Campo receptivo local com filtro de pesos compartilhados <math>5 \times 5</math>, conectado ao primeiro neurônio oculto de um dos mapas de atributos da primeira camada C1.</i>	22
<b>Figura 2.14</b> – <i>Operação de convolução não linear da entrada com um filtro <math>5 \times 5</math>.</i>	23
<b>Figura 2.15</b> – <i>Camada de convolução com <math>k</math> mapas de atributos para detecção de vários tipos de atributos. Cada unidade de um mapa de atributos compartilha os mesmos pesos do filtro correspondente, detectando o mesmo atributo em locais diferentes.</i>	24
<b>Figura 2.16</b> – <i>Convolução de <math>k</math> filtros 3D com a imagem de entrada RGB, gerando um conjunto de <math>k</math> mapas de atributos 2D na saída de uma camada de convolução.</i>	24
<b>Figura 2.17</b> – <i>Convolução com passo unitário (a); convolução com passo 2 (b); sem zero-padding reduz tamanho na saída (c); zero-padding igual a 1 mantém o tamanho da entrada após convolução com filtro <math>3 \times 3</math> e passo 1 (d) [DUMOULIN e VISIN 2016].</i>	25
<b>Figura 2.18</b> – <i>Maxpooling (a) e average-pooling (b) com campo receptivo não sobreposto <math>2 \times 2</math> (passo 2); subamostragem sobreposta com campo <math>3 \times 3</math> e passo 2 (c).</i>	26
<b>Figura 2.19</b> – <i>Arquitetura de rede AlexNet. Adaptado de [ZHANG et al. 2020].</i>	28
<b>Figura 2.20</b> – <i>Mapas de AlexNet distribuídos em duas GPUs. Adaptado de [KRIZHEVSKY et al. 2012].</i>	29
<b>Figura 2.21</b> – <i>Aumento artificial do conjunto de dados de treinamento com transformações de imagens (redimensionamento e corte para <math>256 \times 256</math>, translação com corte aleatório de <math>224 \times 224</math>, reflexão horizontal e alterações de iluminação/cor).</i>	30
<b>Figura 2.22</b> – <i>Previsão final pela combinação das previsões de 10 transformações da imagem de teste, usando o recorte central e os quatro cantos, bem como seus reflexos horizontais.</i>	30
<b>Figura 2.23</b> – <i>Rede composta de um conjunto de redes treinadas independentemente.</i>	31
<b>Figura 2.24</b> – <i>Transformações adicionais de imagens de treinamento, com cortes aleatórios de <math>224 \times 224</math> sobre uma imagem retangular redimensionada (a); recortes de <math>224 \times 224</math> de alta resolução em imagens de tamanho <math>448 \times N</math> (ou <math>N \times 448</math>) (b).</i>	31
<b>Figura 2.25</b> – <i>Transformações nas imagens de teste com a combinação de 5 cortes com translações, 2 espelhamentos, 3 escalas e 3 vistas com total de 90 imagens para previsões.</i>	32
<b>Figura 2.26</b> – <i>Configurações VGG com 11 a 19 camadas [SIMONYAN e ZISSERMAN 2015].</i>	33
<b>Figura 2.27</b> – <i>Convoluções <math>3 \times 3</math> consecutivas com campo receptivo <math>5 \times 5</math> e <math>7 \times 7</math>.</i>	33

<b>Figura 2.28</b> – <i>Mapas de atributos da rede VGG-16 com 16 camadas (exceto maxpooling). Adaptado de [SIMONYAN e ZISSERMAN 2015].</i> .....	34
<b>Figura 2.29</b> – <i>Estrutura paralela do bloco Inception-v1 (a) e mapas de atributos em diferentes escalas concatenados na saída do bloco (b). Adaptado de [ZHANG et al. 2020].</i> .....	35
<b>Figura 2.30</b> – <i>Arquitetura de rede GoogLeNet com módulos Inception. Adaptado de [ZHANG et al. 2020; SZEGEDY et al. 2015].</i> .....	36
<b>Figura 2.31</b> – <i>Outras versões de módulos Inception. Módulo A, onde cada convolução <math>5 \times 5</math> é fatorada em duas convoluções <math>3 \times 3</math>. Módulo B com a fatoração das convoluções <math>n \times n</math> do Módulo A em convoluções <math>n \times 1</math> e <math>1 \times n</math>. Módulo C, expansão da largura (número de canais) com convoluções <math>n \times 1</math> e <math>1 \times n</math> assimétricas paralelas [SZEGEDY et al. 2016].</i> .....	37
<b>Figura 2.32</b> – <i>Módulos residuais Inception-ResNet A, B e C. Adaptado de [SZEGEDY et al. 2017].</i> .....	37
<b>Figura 2.33</b> – <i>Esquema simplificado das redes Inception: substituição dos módulos Inception originais por módulos fatorados A, B e C da Figura 2.31 ou módulo Inception-ResNet A, B e C da Figura 2.32; substituição de maxpooling por módulos redutores de grade; e fatoração da convolução inicial <math>7 \times 7</math> em três convoluções <math>3 \times 3</math>. As variantes do módulo de redução A e B foram omitidas na figura. Baseado em [SZEGEDY et al. 2016; SZEGEDY et al. 2017].</i> .....	38
<b>Figura 2.34</b> – <i>Bloco residual com conexão de atalho para mapeamento de identidade. Adaptado de [ZHANG et al. 2020].</i> .....	39
<b>Figura 2.35</b> – <i>Blocos residuais regulares: bloco de identidade (a) que não altera a dimensão da entrada; bloco convolucional que altera a dimensão espacial da entrada (b). Adaptado de [ZHANG et al. 2020].</i> .....	40
<b>Figura 2.36</b> – <i>Blocos residuais de gargalo: bloco de identidade (a); bloco convolucional (b).</i> .....	40
<b>Figura 2.37</b> – <i>Arquiteturas de ResNet-34 (topo) e ResNet-50 (inferior) [HE et al. 2016a].</i> .....	41
<b>Figura 2.38</b> – <i>Convolução regular captura correlações espaciais e correlações entre canais simultaneamente.</i> .....	43
<b>Figura 2.39</b> – <i>Convoluções <math>1 \times 1</math> seguidas de convoluções regulares usada em um ramo do módulo Inception original.</i> .....	43
<b>Figura 2.40</b> – <i>Módulo Inception original simplificado (a) e módulo Inception simplificado equivalente (b). Versão extrema do módulo Inception: convoluções de canal cruzado (<math>1 \times 1</math>) e convoluções espaciais 2D (<math>3 \times 3</math>) por canal [CHOLLET 2017].</i> .....	44
<b>Figura 2.41</b> – <i>Convoluções do módulo Inception simplificado: convoluções de canal cruzado (<math>1 \times 1</math>) e convoluções regulares 3D (<math>3 \times 3</math>).</i> .....	44
<b>Figura 2.42</b> – <i>Versão extrema do módulo Inception: convoluções de canal cruzado (<math>1 \times 1</math>) e convoluções espaciais 2D (<math>3 \times 3</math>) por canal.</i> .....	44

<b>Figura 2.43</b> – <i>Convoluções separáveis em profundidade: convoluções espaciais 2D (<math>3 \times 3</math>) por canal e convoluções de canal cruzado (<math>1 \times 1</math>).</i> .....	45
<b>Figura 2.44</b> – <i>Fluxo de entrada, intermediário e de saída da arquitetura Xception com módulos residuais de convoluções separáveis em profundidade. Baseado em [CHOLLET 2017].</i> .....	45
<b>Figura 2.45</b> – <i>Mapas de atributos da rede Xception.</i> .....	46
<b>Figura 2.46</b> – <i>Treinamento da rede Xception com/sem conexões residuais [CHOLLET 2017].</i> .....	46
<b>Figura 2.47</b> – <i>Bloco denso de 4 blocos convolucionais, onde cada bloco usa todos os mapas de atributos anteriores como entrada. Taxa de crescimento de <math>k</math> mapas de atributos dentro de um bloco denso. Adaptado de [HUANG et al. 2017].</i> ..	47
<b>Figura 2.48</b> – <i>Tipos de blocos convolucionais DenseNet.</i> .....	48
<b>Figura 2.49</b> – <i>DenseNet com 4 blocos densos e camadas de transição. Adaptado de [HUANG et al. 2017].</i> .....	48
<b>Figura 2.50</b> – <i>Camada de transição entre dois blocos densos.</i> .....	49
<b>Figura 2.51</b> – <i>Comparação do número de parâmetros das redes de classificação. Fonte: <a href="https://keras.io/api/applications/">https://keras.io/api/applications/</a>.</i> .....	51
<b>Figura 2.52</b> – <i>Comparação da profundidade em número de camadas convolucionais das redes de classificação, incluindo camadas de ativação e BN. Fonte: <a href="https://keras.io/api/applications/">https://keras.io/api/applications/</a>.</i> .....	51
<b>Figura 2.53</b> – <i>Comparação de desempenho de classificação no conjunto ImageNet. Fonte: <a href="https://keras.io/api/applications/">https://keras.io/api/applications/</a>.</i> .....	52
<b>Figura 2.54</b> – <i>Segmentação semântica usando CNN para classificação de patches [NOGUEIRA et al. 2016].</i> .....	53
<b>Figura 2.55</b> – <i>Tamanho dos mapas de atributos nos blocos VGG-16 com campo receptivo de <math>224 \times 224</math> (com same padding) e fator de saída de 32.</i> .....	53
<b>Figura 2.56</b> – <i>Categorias de arquiteturas de segmentação semântica de imagens.</i> .....	54
<b>Figura 2.57</b> – <i>Exemplos de imagens do conjunto PASCAL-VOC [EVERINGHAM et al. 2015].</i> .....	55
<b>Figura 2.58</b> – <i>Arquitetura de rede Hipercolumns com os mapas de atributos do backbone AlexNet e camadas convolucionais e de sobreamostragem adicionais (caixas azuis).</i> .....	56
<b>Figura 2.59</b> – <i>Mapas de atributos de VGG-16 com camadas totalmente conectadas.</i> .....	57
<b>Figura 2.60</b> – <i>Convolucionalização das camadas totalmente conectadas.</i> .....	58
<b>Figura 2.61</b> – <i>CNN-base (VGG-16) convertida em rede totalmente convolucional.</i> .....	58
<b>Figura 2.62</b> – <i>Mapas de atributos da rede FCN-32s com rede-base VGG-16.</i> .....	59



<b>Figura 2.63</b> – <i>Convolução transposta para upsampling de 2x da entrada <math>3 \times 3</math> (azul) para uma saída <math>5 \times 5</math> (verde), com kernel <math>3 \times 3</math> e passo <math>s = 2</math> (<math>s - 1 &gt; 0</math>: número de zeros inseridos entre elementos da entrada para obter sobreamostragem de <math>s</math> vezes). Implementação como convolução normal com kernel <math>3 \times 3</math> com passo <math>s = 1</math> [DUMOULIN e VISIN 2016].</i>	60
<b>Figura 2.64</b> – <i>Mapas de atributos da rede FCN-16s com rede-base VGG-16.</i>	60
<b>Figura 2.65</b> – <i>Mapas de atributos da rede FCN-8s com rede-base VGG-16.</i>	61
<b>Figura 2.66</b> – <i>Sequência de deconvoluções com três convoluções transpostas.</i>	61
<b>Figura 2.67</b> – <i>Arquitetura das redes FCN-32s, 16s e 8s com deconvolução transposta.</i>	62
<b>Figura 2.68</b> – <i>Hierarquia de atributos com informação local e global.</i>	62
<b>Figura 2.69</b> – <i>Combinação de informação de atributos local e global [LONG et al. 2015].</i>	63
<b>Figura 2.70</b> – <i>Resultado de segmentação semântica das redes FCN-32s, FCN-16s e FCN-8s de uma imagem do conjunto PASCAL-VOC [LONG et al. 2015].</i>	63
<b>Figura 2.71</b> – <i>Mapas de atributos da rede U-Net [RONNEBERGER et al. 2015].</i>	64
<b>Figura 2.72</b> – <i>Arquitetura codificador-decodificador da rede U-Net com conexão de salto.</i>	65
<b>Figura 2.73</b> – <i>Mapas de atributos da rede DeconvNet. Adaptado de [NOH et al. 2015].</i>	66
<b>Figura 2.74</b> – <i>Operações de unpooling (a) e deconvolution (b)[NOH et al. 2015]. Convolução transposta de mapa esparso (azul) com kernel <math>3 \times 3</math> e <math>s = 1</math> (c) [DUMOULIN e VISIN 2016].</i>	67
<b>Figura 2.75</b> – <i>Arquitetura DeconvNet: redes de convolução e deconvolução. Baseado em [NOH et al. 2015].</i>	67
<b>Figura 2.76</b> – <i>Visualização dos mapas da rede de deconvolução [NOH et al. 2015].</i>	68
<b>Figura 2.77</b> – <i>Arquitetura da rede SegNet-v1. Baseado em [BADRINARAYANAN et al. 2015].</i>	69
<b>Figura 2.78</b> – <i>Arquitetura da rede SegNet-v2. Baseado em [BADRINARAYANAN et al. 2017].</i>	69
<b>Figura 2.79</b> – <i>Contexto espacial de SegNet-v1 com campo receptivo de <math>106 \times 106</math> pixels.</i>	69
<b>Figura 2.80</b> – <i>Mapas de atributos da rede SegNet-v2 com backbone VGG-16 no codificador e um decodificador correspondente. Adaptado de [BADRINARAYANAN et al. 2017].</i>	70
<b>Figura 2.81</b> – <i>Índices de maxpooling para upsampling (sem aprendizado) dos mapas de atributos. Adaptado de [BADRINARAYANAN et al. 2017].</i>	70
<b>Figura 2.82</b> – <i>Arquitetura de rede RefineNet. Adaptado de [LIN, MILAN et al. 2017].</i>	73
<b>Figura 2.83</b> – <i>Mapas de atributos do módulo frontend FCN-32s (VGG-16) modificado, seguido do módulo de contexto da rede DilatedNet.</i>	75

- Figura 2.84** – *Módulo frontend com uma FCN modificada (FCN-32s ou DeepLab-v1), seguido de um módulo de contexto.* ..... 75
- Figura 2.85** – *Configuração de um módulo de contexto da rede DilatedNet. Baseado em [YU e KOLTUN 2016].* ..... 75
- Figura 2.86** – *Arquitetura DilatedNet com frontend FCN-32s modificado e módulo de contexto com convolução atrous. Baseado em [YU e KOLTUN 2016].* ..... 76
- Figura 2.87** – *Convolução atrous 2D com kernel  $3 \times 3$  e diferentes taxas de dilatação  $r = 1, 2$  e  $4$ . Pixels vermelhos denotam as entradas ponderadas pelos pesos do kernel, com pixel de saída no centro. Pixels azuis denotam o campo receptivo. Adaptado de [CHEN et al. 2017].* ..... 77
- Figura 2.88** – *Extração de atributos densos com convolução atrous com kernel  $k = 3$ , taxa de dilatação  $r = 2$  e passo de saída  $= 1$ , aplicada em uma entrada de alta resolução com preenchimento  $p = 2$  (a); Extração de atributos esparsos com convolução padrão,  $k = 3$ , passo de entrada e saída igual a 1, em uma entrada subamostrada de baixa resolução (b); Extração de atributos densos com convolução padrão,  $k = 3$ , passo de entrada e saída 1, em uma entrada de alta resolução e menor campo receptivo ( $FoV = 3$ ) (c); Convolução padrão com kernel grande ( $k = 5$ ), passo de entrada e saída 1, com maior número de parâmetros e campo receptivo ( $FoV = 5$ ) (d). Adaptado de [CHEN et al. 2015; CHEN et al. 2018a].* ..... 77
- Figura 2.89** – *Campos receptivos de uma pilha de convoluções regulares ou atrous.* ..... 78
- Figura 2.90** – *Mapas de atributos da rede DeepLab-v1 básica com convolução atrous.* ... 79
- Figura 2.91** – *Arquitetura da rede DeepLab-v1 básica.* ..... 80
- Figura 2.92** – *Arquitetura da rede DeepLab-MSc com backbone VGG e conexões de salto: os blocos brancos são classificadores MLP (detalhe no quadro tracejado lateral).* ..... 80
- Figura 2.93** – *Arquitetura da rede DeepLab-LargeFoV com grande campo de visão.* ..... 81
- Figura 2.94** – *Pirâmide espacial de agrupamentos (SPP) [HE et al. 2015b] (a); Pirâmide espacial de agrupamentos atrous (ASPP) [CHEN et al. 2018a] (b).* ..... 81
- Figura 2.95** – *Arquitetura da rede DeepLab-v2 com backbone VGG-16 e ASPP-Large. Baseado em [CHEN et al. 2018a].* ..... 82
- Figura 2.96** – *Mapas de atributos de DeepLab-v2 com backbone VGG-16 e ASPP-Large.* ..... 82
- Figura 2.97** – *Arquitetura da rede DeepLab-v2 com backbone ResNet-101 usando ASPP.* ..... 83
- Figura 2.98** – *Blocos em série sem convolução atrous, com fator de redução de 256. Adaptado de [CHEN et al. 2017].* ..... 84
- Figura 2.99** – *DeepLab-v3 com convolução atrous em série, com fator de redução de 16. Adaptado de [CHEN et al. 2017].* ..... 85

<b>Figura 2.100</b> – <i>DeepLab-v3 com convolução atrous (ASPP) em paralelo com concatenação de atributos de contexto global de nível de imagem. Adaptado de [CHEN et al. 2017].</i> .....	85
<b>Figura 2.101</b> – <i>Convolução atrous com taxa de dilatação muito grande: com kernel dilatado maior que entrada, apenas peso central é convoluido de forma semelhante a um kernel padrão <math>1 \times 1</math>.</i> .....	86
<b>Figura 2.102</b> – <i>Módulo de contexto de ParseNet com GAP. Adaptado de [LIU et al. 2015].</i> .....	86
<b>Figura 2.103</b> – <i>Módulo de contexto de PSPNet. Adaptado de [ZHAO et al. 2017].</i> .....	86
<b>Figura 2.104</b> – <i>DeepLab-v3 com backbone ResNet-101 usando ASPP modificado. Baseado em [CHEN et al. 2017].</i> .....	87
<b>Figura 2.105</b> – <i>Módulos em série com convolução atrous, com fator de redução de 8. Baseado em [CHEN et al. 2018b].</i> .....	88
<b>Figura 2.106</b> – <i>DeepLab-v3+ com backbone ResNet-101 usando ASPP e decodificador. Baseado em [CHEN et al. 2018b].</i> .....	89
<b>Figura 2.107</b> – <i>Módulo codificador Xception modificado (Xception-DeepLab) com convolução separável em profundidade atrous. Baseado em [CHEN et al. 2018b].</i> .....	90
<b>Figura 2.108</b> – <i>Convolução separável em profundidade atrous. Adaptado de [CHEN et al. 2018b].</i> .....	90
<b>Figura 2.109</b> – <i>Arquitetura da rede DeepLab-v3+ com backbone Xception-DeepLab. Baseado em [CHEN et al. 2018b].</i> .....	91
<b>Figura 2.110</b> – <i>Arquitetura DenseASPP com backbone DenseNet-121 modificado e módulo DenseASPP, onde a saída <math>d_l</math> de cada bloco convolucional atrous <math>l</math> é concatenada com os atributos dos blocos anteriores, gerando um mapa de atributos <math>[d_{l-1}, d_{l-2}, \dots, d_0]</math> que alimenta o próximo bloco. Adaptado de [YANG et al. 2018].</i> .....	92
<b>Figura 2.111</b> – <i>Mapas de atributos da rede DenseASPP com multiescala densa.</i> .....	93
<b>Figura 2.112</b> – <i>Campos receptivos (<math>FoV = 7</math> e <math>13</math>) de convoluções atrous com taxas <math>r_1 = 3</math> e <math>r_2 = 6</math>. <math>FoV = 19</math> de duas convoluções atrous em série com taxas de dilatação <math>r = (3, 6)</math> (a); Versão 2D da convolução em série (b). Adaptado de [YANG et al. 2018].</i> .....	93
<b>Figura 2.113</b> – <i>DenseASPP com taxas de dilatação (3, 6, 12, 18) produz pirâmide de atributos com maior diversidade de escala (em termos de tamanhos de campos receptivos) e maior campo receptivo do que ASPP. <math>FoV</math> representa o tamanho do campo receptivo da combinação densa correspondente [YANG et al. 2018].</i> .....	94
<b>Figura 2.114</b> – <i>Arquitetura da rede FC-DenseNet103 com 103 camadas convolucionais (a), composto dos seguintes módulos: DB (Dense Block) tem diferentes números de camadas (b); TD (Transition Down) e TU (Transition Up) (c); <math>m</math> corresponde ao número total de mapas de atributos no final de um bloco e <math>c</math> representa o número de classes [JEGOU et al. 2017].</i> .....	95

<b>Figura 2.115</b> – Modelo de CNN com redução do tamanho de entrada (a); poda dos canais da CNN (b); eliminação do último bloco da CNN (c). A linha tracejada preta representa as operações que danificam a informação espacial, enquanto a linha tracejada azul representa as operações que diminuem o campo receptivo. Adaptado de [YU et al. 2018].	97
<b>Figura 2.116</b> – Mapas de atributos do modelo ENet com poda de canais e eliminação do último bloco do backbone. Baseado em [PASZKE et al. 2016].	98
<b>Figura 2.117</b> – Estrutura da rede de segmentação bilateral (BiSeNet) com caminho espacial e caminho de contexto. Adaptado de [YU et al. 2018].	98
<b>Figura 2.118</b> – Mapas de atributos da rede de segmentação bilateral (BiSeNet).	99
<b>Figura 2.119</b> – Diferentes estruturas de redes de segmentação semântica [CHEN et al. 2017; CHEN et al. 2018b].	100
<b>Figura 2.120</b> – Estrutura da rede Attention to Scale [CHEN et al. 2016a].	101
<b>Figura 2.121</b> – Limitações dos algoritmos de segmentação semântica com conexões de salto sem atributos multiescala [NOH et al. 2015].	105
<b>Figura 2.122</b> – Comparação de desempenho de segmentação no conjunto PASCAL-VOC.	107
<b>Figura 2.123</b> – Configurações dos experimentos com ou sem transferência de pesos para modelos de segmentação semântica com backbone semelhante à CNN original (FCN, U-Net, SegNet, RefineNet, BiSeNet) (esquerda) ou backbone com camadas superiores redefinidas (PSPNet, DeepLab-v3, DeepLab-v3+ e DenseASPP), sendo as camadas pré-treinadas (rosa-escuro), ajustadas (rosa-claro) ou inicializadas com pesos aleatórios (azul).	110
<b>Figura 2.124</b> – Resultados de desempenho esperados na comparação do modelo com inicialização aleatória e com transferência de aprendizado. Linha tracejada é o desempenho mIoU do modelo de segmentação com inicialização aleatória. Os pontos vermelho-escuros são os desempenhos do modelo de segmentação usando uma CNN pré-treinada, com n camadas transferidas e congeladas. Os pontos vermelho-claros são as mesmas versões transferidas e ajustadas. Reinterpretado de [YOSINSKI et al. 2014].	111
<b>Figura 2.125</b> – Treinamento em duas etapas para ajuste fino de pesos pré-treinados.	113
<b>Figura 3.1</b> – Mosaico de imagens aéreas com anotação de pixels em quatro classes.	116
<b>Figura 3.2</b> – Espectro de ondas eletromagnéticas das bandas espectrais do sensor RGB (a); espectro de quatro bandas dos sensores multiespectrais (b) [HOLLER et al. 2022].	117
<b>Figura 3.3</b> – Comportamento espectral da vegetação e do solo [HOLLER et al. 2022].	118
<b>Figura 3.4</b> – Mapa de índice de vegetação por diferença normalizada (NDVI).	119
<b>Figura 3.5</b> – Valores de índices de vegetação de cada classe de cobertura de solo (solo exposto, plantas daninhas e cultura). Os valores dos índices são afetados pela altura de voo e tipo de câmera [TORREZ-SÁNCHEZ et al. 2013].	120

- Figura 3.6** – Exemplos de composição colorida e imagem multicanal a partir de imagens multiespectrais e mapas de índice de vegetação. Número de canais em parênteses.....122
- Figura 3.7** – Imagem parcial segmentada usando o algoritmo SLIC [FERREIRA et al. 2017]. .....123
- Figura 3.8** – Exemplos de classes do conjunto de dados de imagens segmentadas. Em cada linha são mostradas quatro amostras de cada classe: ervas daninhas de folhas largas; gramíneas; soja e solo [FERREIRA et al. 2017]......123
- Figura 3.9** – Imagem original e imagem classificada pela CNN. A cultura de soja foi mantida em sua cor original (verde), laranja representa as ervas daninhas de folhas largas, azul, as gramíneas, e magenta, o solo [FERREIRA et al. 2017]. .....124
- Figura 3.10** – Detecção de limite do NDVI extraído das imagens NIR e Red [SA et al. 2017]. .....125
- Figura 3.11** – Vista aérea parcial do campo controlado com níveis variados de herbicida. Uma quantidade máxima de herbicida foi aplicada nas imagens de cultura à direita (retângulo amarelo) e nenhum herbicida foi utilizado nas imagens de ervas daninhas (verde). O centro mostra uma mistura de plantas daninhas e cultura devido ao uso médio de herbicida (vermelho). Mapa de índice NDVI parcial sobre a imagem [SA et al. 2017]......125
- Figura 3.12** – Arquitetura decodificador-decodificador de segmentação semântica densa [SA et al. 2017]. .....126
- Figura 3.13** – Desempenho de 6 modelos por classes (eixo horizontal). Valores mais altos de F1-score indicam o melhor desempenho de detecção [SA et al. 2017]...127
- Figura 3.14** – Resultados qualitativos de duas cenas. As três primeiras colunas são bandas de entrada da rede, com a quarta e quinta coluna mostrando groundtruth e previsões de probabilidade. A probabilidade de cada classe é codificada por cores, onde o fundo, a cultura e a erva daninha são representadas em azul, vermelho e verde, respectivamente [SA et al. 2017]. .....128
- Figura 3.15** – Cinco conjunto de dados de mosaicos multiespectrais. ....128
- Figura 3.16** – Mapeamento de um campo de beterraba de 1.300 m<sup>2</sup>. Cada pirâmide amarela indica a posição onde uma imagem foi tirada, e as linhas verdes são raios entre um ponto 3D e suas múltiplas visualizações [SA et al. 2018]. .....129
- Figura 3.17** – Ortomosaico RGB do conjunto de dados de treinamento #002. A imagem da esquerda mostra o mapa ortomosaico RGB, e as do meio e da direita são partes de cada área em vários níveis de zoom [SA et al. 2018]. .....130
- Figura 3.18** – Tiles do ortomosaico de 12 canais, gerados a partir do conjunto de dados de treinamento #002 com cinco mosaicos multiespectrais, RGB, CIR e NDVI [SA et al. 2018]. .....130

- Figura 3.19** – *Pipeline de processamento. As imagens multiespectrais são primeiro coletadas por VANT e depois processadas para geração de ortomosaicos, que são recortados em tiles para entrada na rede de segmentação semântica [SA et al. 2018].*.....131
- Figura 3.20** – *Avaliação quantitativa da segmentação usando área sob a curva (AUC) do conjunto de dados RedEdge-M. O modelo 5 tem o melhor resultado, o modelo 10 é o SegNet básico com entrada de imagem RGB e o modelo 12 tem apenas uma entrada de imagem NDVI [SA et al. 2018].* .....132
- Figura 3.21** – *Conjunto de dados de teste #003 de um ortomosaico de um campo de beterraba sacarina com GSD de 1 cm/pixel e seu groundtruth correspondente [SA et al. 2018].*.....133
- Figura 3.22** – *Resultado qualitativo de um exemplo de imagem de entrada, groundtruth e previsão de saída, mostrado com um nível de zoom de 500% no mapa de ervas daninhas do ortomosaico RGB do conjunto de dados de teste 003 RedEdge-M [SA et al. 2018].*.....134
- Figura 4.1** – *Etapas do fluxo de trabalho para geração de ortomosaico RGB, mapas de reflectância multiespectrais e mapa NDVI a partir de imagens aéreas de VANT.* .....137
- Figura 4.2** – *Campo experimental de cana-de-açúcar da FEAGRI-Unicamp, onde foi coletado o conjunto de imagens RGB e multiespectrais de VANT. Fonte: Google Earth.*.....138
- Figura 4.3** – *Plantas daninhas de folhas largas encontradas no campo experimental de cana-de-açúcar. (a) Recorte de  $600 \times 450$  pixels de uma das imagens aéreas de VANT, mostrando uma folha de mamona de aproximadamente 40 cm de largura, dentro do retângulo amarelo de  $40 \times 40$  pixels. (b) Mamona (*Ricinus comunis*).* .....138
- Figura 4.4** – *VANT multirrotor quadricóptero DJI Phantom 3 Professional equipado com câmera multiespectral na posição nadir (a); Sensor solar e câmera multiespectral Sequoia com cinco sensores óticos embutidos (RGB e quatro bandas multiespectrais – Red, Green, Red-Edge e Near Infrared) (b).*....139
- Figura 4.5** – *Resolução das imagens dos sensores da câmera multiespectral Sequoia.*...140
- Figura 4.6** – *Fatores que influenciam no GSD e área de cobertura da imagem no solo. Baseado em [BRITO e COELHO 2012; WOLF et al. 2014; JENSEN 2009].*...140
- Figura 4.7** – *Plano de voo do campo experimental de cana-de-açúcar realizado por um VANT, onde cada ponto verde indica a posição geográfica (X, Y, Z) da imagem capturada (a); cada pirâmide azul indica a posição e orientação da câmera onde uma imagem foi capturada, sendo descartadas as pirâmides vermelhas correspondentes à decolagem (b).*.....141
- Figura 4.8** – *Sobreposição frontal e lateral para obtenção de keypoints 2D em cada imagem aérea (a), para reconstrução de pontos 3D no terreno por estereoscopia (b). Baseado em [BRITO e COELHO 2012; JENSEN 2009].*...142
- Figura 4.9** – *Conjuntos de imagens aéreas RGB e Red-Edge capturadas com sobreposição frontal e lateral na mesma trajetória de voo, mostradas sem correção de*

*distorções de perspectiva (a). Visualização das posições geográficas das imagens capturadas (pirâmides azuis) e imagens iniciais do voo descartadas (pirâmides vermelhas) sobre o terreno 3D (b). .....143*

- Figura 4.10** – *Uma imagem de cada banda multiespectral e do espectro visível capturada pela câmera Sequoia. A última imagem apresenta uma visão ampliada da área dentro da caixa vermelha, mostrando detalhes que sugerem os desafios visuais na distinção entre cultura e ervas daninhas em imagens RGB. Fonte: Imagens capturadas pelo CTBE. ....144*
- Figura 4.11** – *GNSS-RTK para obtenção das coordenadas geográficas de alta precisão dos pontos de controle (GCPs) em marcadores distribuídos na área de interesse do terreno. ....145*
- Figura 4.12** – *Deteção de keypoints e correspondência de características. ....146*
- Figura 4.13** – *Geração de mosaico por deteção/correspondência de keypoints e transformações geométricas das imagens com o software Kolor Autopano Giga 4.4. ....146*
- Figura 4.14** – *Etapas de processamento de imagens e saídas geradas no PIX4Dmapper. ....147*
- Figura 4.15** – *(a) Orientação interior da câmera (distância focal, etc.); (b) Orientação exterior das imagens: posição (X, Y, Z) e atitude do VANT (roll, pitch, yaw –  $\omega, \varphi, k$ ). ....147*
- Figura 4.16** – *Poses iniciais das imagens aéreas (azul) e poses ajustadas (verde). ....148*
- Figura 4.17** – *Sobreposição de imagens para extração de keypoints correspondentes em múltiplas imagens e pontos-chave encontrados em uma imagem RGB...148*
- Figura 4.18** – *Geração de nuvem de pontos 3D esparsa usando aerotriangulação (SfM). Tiepoints ( $P_i$ ) são pontos 3D no terreno visualizados em várias imagens, que correspondem aos keypoints homólogos ( $p_i$ ) detectados automaticamente no espaço da imagem. ....149*
- Figura 4.19** – *Múltiplas visualizações de um ponto 3D nas imagens sobrepostas, mostrando feixes de raios e os keypoints correspondentes. ....149*
- Figura 4.20** – *Densificação da nuvem de pontos 3D e triangulação para geração da malha 3D para visualização do modelo 3D do terreno. ....150*
- Figura 4.21** – *Nuvem de pontos 3D densa e malha texturizada 3D sem precisão métrica. ....150*
- Figura 4.22** – *Modelo digital de superfície (MDS) e de terreno (MDT). ....151*
- Figura 4.23** – *Modelos digitais de elevação em formato raster GeoTIFF (a); e malha de pontos 3D gerada por triangulação de Delaunay [WOLF et al. 2014] (b). 151*
- Figura 4.24** – *Distorção radial simétrica e distorção tangencial [WOLF et al. 2014]. ....152*
- Figura 4.25** – *Imagem multiespectrais com maior distorção radial da lente olho-de-peixe. ....153*

<b>Figura 4.26</b> – <i>Distorção causada pelo rolling shutter da câmera em movimento.</i> .....	153
<b>Figura 4.27</b> – <i>Instabilidade do VANT sem estabilizador de câmera (gimbal).</i> .....	154
<b>Figura 4.28</b> – <i>Perda de sobreposição frontal e/ou lateral devido à inclinação da câmera. Adaptado de [WOLF et al. 2014].</i> .....	154
<b>Figura 4.29</b> – <i>Distorção perspectiva de acordo com a inclinação da câmera [WOLF et al. 2014].</i> .....	154
<b>Figura 4.30</b> – <i>Imagens originais com distorções de perspectiva devido à inclinação da câmera e transformação geométrica afim das imagens sobre o ortomosaico 2D.</i> .....	155
<b>Figura 4.31</b> – <i>Tipos de projeção de imagens: (a) projeção perspectiva central cônica a partir de um centro de projeção CP e (b) projeção perspectiva ortogonal a partir do infinito. Baseado em [BRITO e COELHO 2012].</i> .....	155
<b>Figura 4.32</b> – <i>(a) Deslocamento do “relevo” causado pela projeção cônica em duas alturas de voo; (b) Distorção radial com deslocamento a partir do centro da imagem. Baseado em [WOLF et al. 2014].</i> .....	156
<b>Figura 4.33</b> – <i>Geração de uma ortomagem verdadeira. (a) Duas imagens geram oclusões na ortofoto verdadeira. (b) Três imagens contribuem com a ortofoto verdadeira sem oclusões devido aos diferentes ângulos de visão.</i> .....	156
<b>Figura 4.34</b> – <i>Escala de uma imagem sobre um terreno plano (a); Variação de escala de objetos de uma imagem aérea devido ao relevo irregular (b). Adaptado de [WOLF et al. 2014; JENSEN 2009].</i> .....	157
<b>Figura 4.35</b> – <i>Variação de escala devido ao relevo irregular (a); redução da sobreposição frontal (b) e lateral (c). Adaptado de [WOLF et al. 2014].</i> .....	157
<b>Figura 4.36</b> – <i>Projeção perspectiva central cônica (retângulo verde vazio) e projeção ortogonal (retângulo cheio) para ortorretificação da imagem sobre um relevo (a); projeção ortogonal sobre MDE a partir de múltiplas imagens para gerar ortomosaico verdadeiro (b). Adaptado de [JENSEN 2009].</i> .....	158
<b>Figura 4.37</b> – <i>Detalhes das imagens originais e mesclagem dos pixels para gerar uma parte do ortomosaico RGB.</i> .....	159
<b>Figura 4.38</b> – <i>(a) Blending do mosaico; (b) Editor de mosaico do PIX4Dmapper que permite seleção da imagem a ser projetada no polígono selecionado [PIX4DMAPPER].</i> .....	159
<b>Figura 4.39</b> – <i>Alvo de calibração radiométrica.</i> .....	161
<b>Figura 4.40</b> – <i>Esquema geral de processamento de imagens aéreas para geração da nuvem de pontos 3D, malha de pontos 3D, modelo de superfície 2½D, ortomosaicos 2D georreferenciados, mapas de reflectância 2D e mapas de índice de vegetação 2D.</i> .....	163
<b>Figura 4.41</b> – <i>Imagens de entrada e produtos de saída do PIX4Dmapper [PIX4DMAPPER].</i> .....	164



- Figura 4.42** – *Número de imagens sobrepostas para cada pixel do mosaico RGB e multiespectral. Maior número de imagens sobrepostas melhora a precisão [PIX4DMAPPER]*. .....164
- Figura 4.43** – *Problemas na etapa de calibração, gerando falhas no processamento da nuvem de pontos. Solução com diminuição da escala da imagem para cálculo de keypoints*. .....165
- Figura 4.44** – *Nuvens de pontos 3D densificados dos projetos RGB e multiespectral*. ...165
- Figura 4.45** – *Modelos digitais de superfície (MDS) de baixa resolução do conjunto de dados do campo experimental de cana-de-açúcar*. .....166
- Figura 4.46** – *Ortomosaicos RGB de  $8.196 \times 8.260$  pixels, com GSD médio de  $1,16$  cm/pixel, gerado a partir de 48 imagens RGB de  $4.608 \times 3.456$  pixels*. .....167
- Figura 4.47** – *Detalhes do ortomosaico RGB com GSD de  $1,16$  cm/pixel. A imagem esquerda mostra o mapa ortomosaico de  $8.196 \times 8.260$  pixels. As imagens dentro das caixas coloridas mostram uma ampliação de cada área em vários níveis de zoom. A caixa preta com duas folhas de mamona tem  $30 \times 30$  pixels, que corresponde a  $34,8$  cm*. .....167
- Figura 4.48** – *Detalhe da imagem original 56 (a); mesclagem dos pixels das imagens 5, 56 e 57 para gerar o mosaico RGB dentro do retângulo roxo da Figura 4.47 (b)*. .....168
- Figura 4.49** – *Mapas de reflectância e máscaras das quatro bandas espectrais (Green, Red, Red-Edge e NIR) com  $2.470 \times 2.903$  pixels cada, cobrindo uma área de  $0,013$  km<sup>2</sup>, com GSD de  $4,63$  cm/pixel, gerado a partir de 4 x 48 imagens multiespectrais de  $1.280 \times 960$  pixels*. .....169
- Figura 4.50** – *Detalhes do mapa de reflectância Red-Edge com GSD de  $4,63$  cm/pixel. A imagem esquerda mostra o mapa de  $2.470 \times 2.903$  pixels. As caixas coloridas ampliam cada área em vários níveis de zoom. A caixa preta com folha de mamona tem  $8 \times 8$  pixels, que corresponde a  $37$  cm*. .....169
- Figura 4.51** – *Mapa de índice NDVI visualizado em tons de cinza, calculado com base nos mapas de reflectância NIR e Red*. .....170
- Figura 4.52** – *Histograma do mapa de índice NDVI e paletas de cores para visualização em falsa-cor*. .....170
- Figura 4.53** – *Mapa de índice NDVI calculado e visualizado em falsa-cor no QGIS*. ....171
- Figura 4.54** – *Ortomosaico RGB e mapa de índice VARI, calculado com base nas bandas espectrais visíveis RGB, visualizado em tons de cinza ou paleta de cores*. .....171
- Figura 4.55** – *Recorte do ortomosaico RGB e mapa Red-Edge, com mapas de índices visualizados em alto contraste com diferentes especificações das paletas de cores*. .....172
- Figura 4.56** – *Composição colorida das bandas espectrais 2, 4, 3 (Red, NIR, Red-Edge)*. .....173

<b>Figura 4.57</b> – Exemplos de composições coloridas multiespectrais nos canais RGB. ....	174
<b>Figura 4.58</b> – Mosaicos RGB e mapa de reflectância NIR desalinhados. ....	174
<b>Figura 4.59</b> – Medidas dos deslocamentos entre as bandas Green, Red, Red-Edge e NIR devido à posição das lentes na câmara multiespectral Sequoia. ....	175
<b>Figura 4.60</b> – Desalinhamento espacial entre as bandas multiespectrais Green, Red, Red-Edge e NIR nas imagens originais (a); Mapas de reflectância multiespectrais alinhados. ....	175
<b>Figura 4.61</b> – (a) Estimativa de movimento de câmara com rolling shutter. A linha verde mostra o plano de voo e as posições das imagens. Os pontos azuis representam a posição da câmara no início da exposição. As linhas azuis representam o movimento da câmara durante a leitura do obturador rolante; (b) Imagens RGB capturadas com sensor do tipo global shutter. ....	176
<b>Figura 4.62</b> – Deslocamentos entre ortomosaico RGB e mapa de reflectância NIR. ....	177
<b>Figura 5.1</b> – Etapas do fluxo de trabalho para avaliação das redes de segmentação. ....	179
<b>Figura 5.2</b> – Esquema geral para treinamento das redes de segmentação a partir das imagens aéreas originais e teste das redes a partir do mosaico RGB. Os groundtruths e predições mostram os rótulos das classes solo, cultura, daninha e gramínea em cores. ....	180
<b>Figura 5.3</b> – 48 imagens RGB calibradas (a) e 12 imagens selecionadas para treinamento (verde), validação (azul) e teste (lilás) das redes de segmentação. ....	181
<b>Figura 5.4</b> – Rotulação manual de 12 imagens aéreas RGB sobrepostas de $4.608 \times 3.456$ pixels, com seus groundtruths contendo quatro classes: solo, cultura, daninha e gramínea. ....	182
<b>Figura 5.5</b> – Conjunto de dados particionado em treinamento, validação e teste, composto de imagens originais e seus groundtruths, divididos em uma grade regular de $8 \times 6$ blocos de $576 \times 576$ pixels. ....	183
<b>Figura 5.6</b> – Detalhes de quatro blocos da imagem 01 e groundtruths de $576 \times 576$ pixels. ....	183
<b>Figura 5.7</b> – Hierarquia de pastas do conjunto de dados de treinamento, validação e teste. ....	184
<b>Figura 5.8</b> – Rotulação manual dos pixels do mosaico, usando software de edição de imagens, para teste das redes de segmentação semântica. ....	185
<b>Figura 5.9</b> – Groundtruths de 8 e 24 bits com 4 camadas transparentes sobre o mosaico. ....	186
<b>Figura 5.10</b> – Sobreposição das imagens aéreas no mosaico (a); groundtruth colorido de 24 bits do mosaico RGB com $8.196 \times 8.260$ pixels. ....	186
<b>Figura 5.11</b> – Conjunto de dados de teste: ortomosaico RGB, anotado e expandido para $8.704 \times 8.704$ pixels, particionado em $17 \times 17$ blocos de $512 \times 512$ pixels, para teste final das redes de segmentação semântica. ....	187

<b>Figura 5.12</b> – Hierarquia de pastas do conjunto de teste do mosaico RGB. ....	188
<b>Figura 5.13</b> – Histograma dos mapas de reflectância multiespectrais e mapa NDVI. ...	189
<b>Figura 5.14</b> – Histogramas dos mosaicos multiespectrais multicanais GeoTIFF em ponto flutuante (a) e convertido para PNG de 24 bits (b). ....	189
<b>Figura 5.15</b> – Mosaicos multiespectrais multicanais em formato GeoTIFF de três canais (tipo float de 32 bits), visualizados como composição colorida no QGIS. ....	190
<b>Figura 5.16</b> – Mosaicos multiespectrais em formato PNG com três canais de 8 bits. ....	190
<b>Figura 5.17</b> – Rotulação manual dos mapas multiespectrais multicanais. ....	191
<b>Figura 5.18</b> – Mosaico multiespectral multicanal e seu groundtruth colorido de $2.560 \times 3.072$ pixels, com preenchimento nas bordas, particionados em $10 \times 12$ blocos de $256 \times 256$ pixels, para treinamento, validação e teste das redes de segmentação semântica. ....	192
<b>Figura 5.19</b> – Blocos do conjunto de treinamento, validação e teste. ....	193
<b>Figura 5.20</b> – Hierarquia de pastas do conjunto de teste dos mapas multiespectrais. ....	193
<b>Figura 5.21</b> – Esquema de treinamento/validação e teste das redes de segmentação, a partir da grade de $8 \times 6$ blocos de $576 \times 576$ pixels das imagens originais RGB e seus groundtruths correspondentes. ....	196
<b>Figura 5.22</b> – Processo de treinamento/validação dos modelos de segmentação semântica. ....	197
<b>Figura 5.23</b> – Esquema de geração de blocos de dados para treinamento das redes. ....	198
<b>Figura 5.24</b> – Esquema de geração de blocos de dados para teste das redes treinadas. ....	199
<b>Figura 5.25</b> – Processo de teste dos modelos de segmentação semântica a partir do conjunto de blocos de teste de três imagens RGB originais. ....	199
<b>Figura 5.26</b> – Processo de predição de 8 e 24 bits a partir do conjunto de blocos de teste de três imagens RGB originais. ....	200
<b>Figura 5.27</b> – Processo de avaliação dos modelos de segmentação semântica pela comparação das predições de teste de 8 bits com os blocos groundtruths correspondentes. ....	201
<b>Figura 5.28</b> – Processo de predição de 8 e 24 bits dos modelos de segmentação semântica a partir do conjunto de blocos de teste do mosaico RGB. ....	202
<b>Figura 5.29</b> – Processo de avaliação dos modelos de segmentação semântica pela comparação das predições de 8 bits dos blocos de teste do mosaico RGB e os groundtruths correspondentes. ....	203
<b>Figura 5.30</b> – Composição dos $17 \times 17$ blocos preditos coloridos de $256 \times 256$ pixels, com mascaramento de borda para comparação com o groundtruth. ....	204
<b>Figura 5.31</b> – Tensorboard: desempenho mIoU de treinamento e validação (continuação). ....	207

<b>Figura 5.32</b> – <i>Comparação de desempenho de teste das diferentes configurações de rede usando aumento de dados (DA), inicialização aleatória de pesos (A), transferência sem ajuste dos pesos (NT) ou transferência com ajuste de todas os pesos (AT).</i> .....	210
<b>Figura 5.33</b> – <i>Desempenho mIoU da melhor configuração (AT/DA) de cada modelo no conjunto de imagens aéreas de validação e teste.</i> .....	212
<b>Figura 5.34</b> – <i>Desempenho mIoU por classe e mIoU médio (linha tracejada) da melhor configuração (AT/DA) de cada modelo no conjunto de imagens aéreas de teste.</i> .....	213
<b>Figura 5.35</b> – <i>Relação dos recortes subamostrados para obter diferentes GSDs.</i> .....	213
<b>Figura 5.36</b> – <i>Blocos de teste com diferentes tamanhos e GSDs.</i> .....	214
<b>Figura 5.37</b> – <i>Comparação de desempenho das redes com diferentes resoluções de entrada.</i> .....	215
<b>Figura 5.38</b> – <i>Comparação das quatro redes de melhor desempenho em todas as resoluções.</i> .....	215
<b>Figura 5.39</b> – <i>Tempo de inferência (segundos) e desempenho mIoU (%) no conjunto de três imagens aéreas de teste (144 blocos de 576 » 256).</i> .....	216
<b>Figura 5.40</b> – <i>Tensorboard: desempenho mIoU de treinamento e validação da rede BiSeNet (Xception) AT/DA com diferentes tamanhos de entrada e batch.</i> .....	219
<b>Figura 5.41</b> – <i>Comparação da rede BiSeNet treinada com tamanho de batch diferentes.</i> .....	220
<b>Figura 5.42</b> – <i>Comparação da rede BiSeNet treinada em diferentes tamanhos de blocos.</i> .....	221
<b>Figura 5.43</b> – <i>Comparação do groundtruth com a predição da imagem 10 de teste, prevista pelos modelos de segmentação treinados e testados com blocos 576 » 256 (2 x GSD).</i> .....	222
<b>Figura 5.44</b> – <i>Exemplos de previsão da rede BiSeNet (AT/DA) a partir de quatro blocos rotulados das imagens de teste 10 e 11, mostrando: (a) classificação correta; (b) subestimação de ervas daninhas; (c) falso negativo (não detecção de ervas daninhas); (d) falso positivo (superestimação de ervas daninhas); (e) erro na rotulação manual.</i> .....	224
<b>Figura 5.45</b> – <i>Exemplos de predições que falharam em blocos da imagem original.</i> .....	225
<b>Figura 5.46</b> – <i>Comparação do groundtruth com a predição da imagem 10 de teste, gerada pelos modelos de segmentação treinados com blocos 576 » 256 (2 x GSD) e testadas com blocos de 576 » 512 (1 x GSD).</i> .....	226
<b>Figura 5.47</b> – <i>Modelo BiSeNet treinado com blocos 576 » 256 (2 x GSD) ou blocos 576 » 512 (1 x GSD), e testado com blocos 1.152 » 256 (4 x GSD).</i> .....	228
<b>Figura 5.48</b> – <i>Modelo BiSeNet treinado com 576 » 256 (2 x GSD).</i> .....	229

<b>Figura 5.49</b> – Modelo BiSeNet treinado com 576 » 512 (1 x GSD).....	230
<b>Figura 5.50</b> – Desempenho IoU por classe das melhores configurações (AT/DA) de cada modelo no conjunto de teste do mosaico. ....	231
<b>Figura 5.51</b> – Predições sem máscara de borda, a partir dos blocos 512 » 256 (2 x GSD) do mosaico RGB, com redes treinadas com blocos de imagens aéreas 576 » 256 (2,25 x GSD).....	233
<b>Figura 5.52</b> – Bloco <sub>5,11</sub> do mosaico RGB; groundtruth de 512 × 512 pixels; e predições de 256 × 256 pixels das redes na configuração AT/DA, treinadas com blocos de imagens aéreas 576 » 256 (2,25 x GSD). ....	234
<b>Figura 5.53</b> – Predições do mosaico de tamanho 8.704 × 8.704 pixels, sem máscara de borda, a partir dos blocos de teste 512 » 512 (1 x GSD) do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento 576 » 256 (2,25 x GSD) das imagens aéreas RGB (continuação). ....	235
<b>Figura 5.54</b> – Predição do mosaico sem máscara de borda, a partir dos blocos de treinamento e teste do mosaico RGB com diferentes tamanhos e GSD de entrada.....	237
<b>Figura 5.55</b> – Predição do mosaico sem máscara de borda, a partir dos blocos de teste 512 » 256 (2 x GSD) do mosaico RGB, com diferentes tamanhos de batch. ....	238
<b>Figura 5.56</b> – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelos BiSeNet treinado com blocos de treinamento de tamanho 576 » 256 (2,25 x GSD) das imagens aéreas RGB.....	240
<b>Figura 5.57</b> – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB. ....	242
<b>Figura 6.1</b> – Fluxo de informação dos mapas de atributos nos blocos convolucionais. A altura representa dimensão espacial e a largura representa a quantidade de canais nos mapas de atributos.....	246
<b>Figura 6.2</b> – Modelo A1: modelo de referência. ....	248
<b>Figura 6.3</b> – Modelo A2: modelo de referência com atributos multiescala. ....	249
<b>Figura 6.4</b> – Comparação de desempenho dos modelos A1 e A2. ....	249
<b>Figura 6.5</b> – Modelo B: Modelo com bloco de convolução denso paralelo à CNN. ....	250
<b>Figura 6.6</b> – Desempenho do modelo B com bloco de convolução denso paralelo à CNN. ....	251
<b>Figura 6.7</b> – Modelo C: Modelo com bloco de convolução denso e concatenação das camadas intermediárias da CNN. ....	252
<b>Figura 6.8</b> – Desempenho do modelo C com concatenação das camadas intermediárias. ....	252

<b>Figura 6.9</b> – <i>Modelo D: Modelo com bloco de convolução denso e maior contexto.</i> .....	253
<b>Figura 6.10</b> – <i>Desempenho do modelo D com bloco denso de maior contexto semântico.</i> .....	254
<b>Figura 6.11</b> – <i>Desempenho do modelo de referência com aumento de dados (zoom).</i> .....	254
<b>Figura 6.12</b> – <i>Modelo E: Modelo sem blocos intermediários no bloco denso.</i> .....	255
<b>Figura 6.13</b> – <i>Modelo F: Modelo com bloco denso inverso.</i> .....	256
<b>Figura 6.14</b> – <i>Desempenho do modelo E sem atributos intermediários no bloco denso.</i> ..	257
<b>Figura 6.15</b> – <i>Desempenho do modelo F e DenseASPP com bloco denso no topo da CNN.</i> .....	257
<b>Figura 6.16</b> – <i>Tensorboard: desempenho mIoU de treinamento e validação.</i> .....	260
<b>Figura 6.17</b> – <i>Desempenhos de cada modelo no conjunto de validação e teste.</i> .....	261
<b>Figura 6.18</b> – <i>Comparação de desempenho das redes com diferentes resoluções de entrada.</i> .....	262
<b>Figura 6.19</b> – <i>Modelo proposto com bloco denso paralelo ao backbone.</i> .....	263
<b>Figura 6.20</b> – <i>Mapas de atributos do bloco denso no modelo proposto.</i> .....	264
<b>Figura 6.21</b> – <i>Pirâmide de escalas do mapa de alta resolução do modelo proposto.</i> .....	266
<b>Figura 7.1</b> – <i>Desempenho do modelo proposto, comparado aos modelos multiescalas, no conjunto de validação e teste das imagens aéreas.</i> .....	270
<b>Figura 7.2</b> – <i>Desempenhos mIoU por classe, no conjunto de teste composto por blocos de três imagens aéreas RGB.</i> .....	271
<b>Figura 7.3</b> – <i>Tempo de inferência (segundos) e desempenho mIoU (%) no conjunto de três imagens aéreas de teste (144 blocos de 576 » 256).</i> .....	272
<b>Figura 7.4</b> – <i>Comparação do modelo proposto com as redes de melhor desempenho.</i> ....	273
<b>Figura 7.5</b> – <i>Comparação do modelo proposto com a rede BiSeNet.</i> .....	274
<b>Figura 7.6</b> – <i>Comparação das previsões dos modelos de segmentação a partir de três imagens RGB de teste.</i> .....	276
<b>Figura 7.7</b> – <i>Blocos <math>x</math>, <math>y</math> (<math>n</math>) da imagem <math>n</math> de teste original RGB e seus groundtruths. Previsões das redes BiSeNet, DeepLab-v3+ e modelo proposto, a partir de blocos da imagem original (continuação).</i> .....	277
<b>Figura 7.8</b> – <i>Desempenho IoU por classe dos modelos no conjunto de teste do mosaico.</i> 279	
<b>Figura 7.9</b> – <i>Predição do mosaico de tamanho <math>4.352 \times 4.352</math> pixels, sem máscara de borda, a partir dos blocos de teste 512 » 256 do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento 576 » 256 (2,25 x GSD) das imagens aéreas RGB.</i> .....	280

- Figura 7.10** – *Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelo proposto treinado com blocos de treinamento de tamanho 576 » 256 (2, 25 x GSD) das imagens aéreas RGB.* .....281
- Figura 7.11** – *Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelo proposto treinado com blocos de treinamento de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB.* .....282
- Figura 7.12** – *Comparação da predição a partir de blocos de teste de 2,8 x GSD e 4,25 x GSD do mosaico RGB, usando modelo proposto e BiSeNet treinados com blocos de tamanho 576 » 256 (2,25 x GSD) das imagens aéreas RGB.* .....283
- Figura 7.13** – *Comparação da predição a partir de blocos de teste de 2,8 x GSD e 4,25 x GSD do mosaico RGB, usando modelo proposto e BiSeNet treinados com blocos de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB.* .....284
- Figura 7.14** – *Predições dos blocos 256 × 256 pixels sem subamostragem do mosaico RedEdge–Green–NIR, gerados pelas redes BiSeNet e modelo proposto, treinadas com blocos de 256 × 256 pixels sem subamostragem.* .....286
- Figura 7.15** – *Blocos preditos de 256 × 256 pixels sem subamostragem do mosaico multiespectral composto RedEdge–Green–NIR, gerados pelas redes BiSeNet e rede proposta, treinadas com blocos de 256 × 256 pixels sem subamostragem.* .....287
- Figura 8.1** – *Ortomosaico RGB (SugarcaneWeed) com borda expandida para 5.632 × 7.168 pixels com GSD médio aproximado de 5 cm/pixel [MONTEIRO e WANGENHEIM 2019].* .....290
- Figura 8.2** – *Detalhes de um bloco com diferentes tamanhos e GSDs, com seus respectivos groundtruths com rótulos de classe solo, cultura de cana e daninha.* .....290
- Figura 8.3** – *Ortomosaico RGB (sem plantas daninhas) cortado para 9.216 × 6.144 pixels com GSD médio aproximado de 5 cm/pixel [PEREIRA JUNIOR e WANGENHEIM 2019].* .....291
- Figura 8.4** – *Detalhes de um bloco de 512 × 512 pixels, com maior quantidade de amostras de mudas de cana-de-açúcar, e seu groundtruth com rótulos de classe solo e cana.* .....291
- Figura 8.5** – *Ortomosaico RGB SugarcaneWeed e groundtruth de 5.632 × 7.168 pixels divididos em uma grade regular de 11 x 14 blocos de 512 × 512 pixels.* .293
- Figura 8.6** – *Ortomosaico RGB Sugarcane e groundtruth de 9.216 × 6.144 divididos em uma grade regular de 18 x 12 blocos de 512 × 512 pixels.* .....293
- Figura 8.7** – *Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos 512 » 512 (1 x GSD) dos mosaicos RGB com GSD de 5 cm/pixel.* .....294

- Figura 8.8** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos  $512 \times 512$  ( $1 \times \text{GSD}$ ) dos mosaicos RGB com GSD de 5 cm/pixel.....295
- Figura 8.9** – Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento com blocos de  $512 \times 512$  ( $1 \times \text{GSD}$ ) dos dois mosaicos RGB com GSD de 5 cm/pixel e teste com blocos de  $512 \times 256$  ( $2 \times \text{GSD}$ ) do mosaico SugarcaneWeed. ....296
- Figura 8.10** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento com blocos de  $512 \times 512$  ( $1 \times \text{GSD}$ ) dos dois mosaicos RGB com GSD de 5 cm/pixel e teste com blocos de  $512 \times 256$  ( $2 \times \text{GSD}$ ) de SugarcaneWeed. 297
- Figura 8.11** – Groundtruths do ortomosaico SugarcaneWeed dividido em uma grade regular de  $22 \times 28$  blocos de  $256 \times 256$  pixels. ....298
- Figura 8.12** – Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos de  $256 \times 256$  ( $1 \times \text{GSD}$ ) do mosaico SugarcaneWeed com GSD de 5 cm/pixel. ....299
- Figura 8.13** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos de  $256 \times 256$  ( $1 \times \text{GSD}$ ) do mosaico SugarcaneWeed com GSD de 5 cm/pixel. ....300
- Figura A.1** – Seleção de 12 das 48 imagens RGB originais calibradas de  $4.608 \times 3.456$  pixels para formar o conjunto de treinamento (vermelho), validação (azul) e teste (amarelo). ....307
- Figura A.2** – As 12 imagens RGB de  $4.608 \times 3.456$  pixels selecionadas para treinamento (vermelho), validação (azul) e teste (amarelo). ....309
- Figura A.3** – Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste. ..310
- Figura A.4** – Conjunto de teste com  $17 \times 17$  blocos de  $512 \times 512$  pixels do mosaico RGB, dos quais 100 blocos nas bordas apresentam fundo preto (valor de pixel zero). ....316
- Figura A.5** – Conjunto de teste com  $17 \times 17$  blocos de  $512 \times 512$  pixels de groundtruths do mosaico RGB, dos quais 100 blocos nas bordas (de fundo preto) não são válidos para medida de desempenho. ....317
- Figura A.6** – Ortomosaico RGB expandido com preenchimento nas bordas e groundtruth, com  $8.704 \times 8.704$  pixels para teste das redes de segmentação semântica. ....318
- Figura A.7** – Mapas multiespectrais multicanais e groundtruths com preenchimento nas bordas, com  $2.560 \times 3.072$  pixels para treinamento e teste das redes de segmentação semântica. A banda NIR ou RedEdge foi usada como referência para rotulação manual dos mapas multiespectrais. ....319
- Figura A.8** – Conjunto de dados com  $10 \times 12$  blocos de  $256 \times 256$  pixels do mapa RedEdge–Grren–NIR, dos quais blocos inválidos nas bordas apresentam fundo preto (valor zero). Conjunto de treinamento com 44 blocos válidos (dentro das



	<i>áreas retangulares verde), validação com 16 blocos (azul), e teste com 15 blocos (lilás).</i> .....	320
<b>Figura A.9</b>	<i>– Conjunto de dados com <math>10 \times 12</math> blocos de <math>256 \times 256</math> pixels de groundtruths do mapa RedEdge–Green–NIR, dos quais blocos com fundo preto são descartados para treinamento e medição de desempenho. Conjunto de treinamento com 44 blocos válidos (dentro das áreas retangulares verde), validação com 16 blocos (azul), e teste com 15 blocos (lilás).</i> .....	321
<b>Figura B.1</b>	<i>– Modelo FCN32s, 16s e 8s com backbone VGG-16.</i> .....	325
<b>Figura B.2</b>	<i>– Modelo UNet com backbone VGG-16.</i> .....	326
<b>Figura B.3</b>	<i>– Modelo SegNet com backbone VGG-16.</i> .....	327
<b>Figura B.4</b>	<i>– Modelo RefineNet com backbone ResNet-50.</i> .....	328
<b>Figura B.5</b>	<i>– Modelo BiSegNet com backbone Xception.</i> .....	329
<b>Figura B.6</b>	<i>– Modelo PSPNet com backbone ResNet-50.</i> .....	330
<b>Figura B.7</b>	<i>– Modelo DeepLab-v3 com backbone ResNet-50.</i> .....	331
<b>Figura B.8</b>	<i>– Modelo DeepLab-v3+ com backbone ResNet-50.</i> .....	332
<b>Figura B.9</b>	<i>– Modelo DeepLab-v3+ com backbone Xception-DeepLab e <math>r = [1, 2]</math>.</i> .....	333
<b>Figura B.10</b>	<i>– Backbone Xception (sem convolução atrous).</i> .....	333
<b>Figura B.11</b>	<i>– Modelo DenseASPP com backbone DenseNet-121.</i> .....	334
<b>Figura B.12</b>	<i>– Modelo DenseASPP com backbone ResNet-50.</i> .....	335
<b>Figura C.1</b>	<i>– Matriz de confusão.</i> .....	337
<b>Figura C.2</b>	<i>– Matriz de confusão de um modelo de classificação multiclasse.</i> .....	339
<b>Figura C.3</b>	<i>– Detecção de objetos com caixa delimitadora.</i> .....	340
<b>Figura C.4</b>	<i>– Intersecção sobre união.</i> .....	341
<b>Figura C.5</b>	<i>– Intersecção sobre união.</i> .....	341
<b>Figura C.6</b>	<i>– Intersecção sobre união.</i> .....	341
<b>Figura C.7</b>	<i>– Segmentação semântica de imagens: máscara de groundtruth em vermelho e máscara predita em ciano.</i> .....	342
<b>Figura D.1</b>	<i>– Camada softmax das redes de classificação.</i> .....	343
<b>Figura D.2</b>	<i>– Convolução transposta [DUMOULIN e VISIN 2016].</i> .....	344



## Lista de tabelas

<b>Tabela 2.1</b> – <i>Arquiteturas ResNet. Os blocos residuais são mostrados entre colchetes. O número de blocos <math>n</math> é indicado ao lado do colchete para cada módulo conv-<math>m_x</math>. A redução da resolução espacial é realizada pelo bloco residual convolucional em conv3_1, conv4_1 e conv5_1 com passo 2 [HE et al. 2016a].</i>	41
<b>Tabela 2.2</b> – <i>Arquiteturas DenseNet para ImageNet, com taxa de crescimento <math>k = 32</math> e camada “conv” como uma sequência BN-ReLU-Conv [HUANG et al. 2017].</i>	49
<b>Tabela 2.3</b> – <i>Configuração da rede DeconvNet [NOH et al. 2015].</i>	68
<b>Tabela 2.4</b> – <i>Tabela comparativa de características e desempenho (mIoU%) das redes de segmentação semântica no conjunto PASCAL-VOC e Cityscapes.</i>	106
<b>Tabela 3.1</b> – <i>Alguns exemplos de índices de vegetação.</i>	119
<b>Tabela 3.2</b> – <i>Combinação de bandas para diferentes estratégias de classificação.</i>	121
<b>Tabela 3.3</b> – <i>Resumo da avaliação de desempenho com canais de entrada variados usando a câmera RedEdge-M [SA et al. 2018].</i>	132
<b>Tabela 4.1</b> – <i>Especificação dos sensores da câmera Sequoia.</i>	139
<b>Tabela 4.2</b> – <i>Conjuntos de imagens aéreas do campo experimental de cana-de-açúcar.</i>	142
<b>Tabela 4.3</b> – <i>Informações do ortomosaico RGB e mapas multiespectrais.</i>	166
<b>Tabela 4.4</b> – <i>Mapa multicanal com cinco bandas espectrais e índice de vegetação NDVI.</i>	173
<b>Tabela 5.1</b> – <i>Conjuntos de dados de treinamento, validação e teste das imagens originais.</i>	184
<b>Tabela 5.2</b> – <i>Conjunto de dados de teste sobre mosaico RGB.</i>	187
<b>Tabela 5.3</b> – <i>Conjunto de dados com blocos do mapa multicanal.</i>	192
<b>Tabela 5.4</b> – <i>Modelos de redes de segmentação semântica com diferentes backbones. ...</i>	194
<b>Tabela 5.5</b> – <i>Diferentes configurações das redes de segmentação semântica baseadas em CNN, número de parâmetros e tempo de treinamento.</i>	206
<b>Tabela 5.6</b> – <i>Desempenho IoU médio das redes de segmentação semântica no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada..</i>	211

<b>Tabela 5.7</b> – <i>Desempenho de mIoU por classe e mIoU médio das redes de segmentação semântica no conjunto de teste composto por blocos de três imagens aéreas RGB.</i> .....	212
<b>Tabela 5.8</b> – <i>Desempenho de teste das redes (AT/DA) com diferentes GSDs de entrada.</i> .....	214
<b>Tabela 5.9</b> – <i>Tempo de execução dos testes das redes com diferentes resoluções de entrada.</i> .....	216
<b>Tabela 5.10</b> – <i>Treinamento da rede BiSeNet AT(5)/DA com diferentes estratégias.</i> .....	217
<b>Tabela 5.11</b> – <i>Desempenho de BiSeNet AT(5)/DA com diferentes estratégias de treinamento.</i> .....	218
<b>Tabela 5.12</b> – <i>Desempenho mIoU de cada estratégia de treinamento da rede BiSeNet AT(5)/DA em diferentes resoluções dos blocos de teste.</i> .....	220
<b>Tabela 5.13</b> – <i>Desempenho das melhores configurações de redes de segmentação semântica por classe no conjunto de teste composto por blocos do ortomosaico RGB.</i>	231
<b>Tabela 6.1</b> – <i>Número de parâmetros e tempo de treinamento dos modelos propostos.</i> ...	258
<b>Tabela 6.2</b> – <i>Desempenho IoU médio dos modelos propostos no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada.</i> .....	261
<b>Tabela 6.3</b> – <i>Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.</i> .....	262
<b>Tabela 7.1</b> – <i>Número de parâmetros e tempo de treinamento dos modelos.</i> .....	269
<b>Tabela 7.2</b> – <i>Desempenho IoU médio dos modelos no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada.</i> .....	270
<b>Tabela 7.3</b> – <i>Desempenho de mIoU por classe, no conjunto de teste composto por blocos de três imagens aéreas RGB.</i> .....	271
<b>Tabela 7.4</b> – <i>Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.</i> .....	272
<b>Tabela 7.5</b> – <i>Tempo de execução dos testes das redes com diferentes resoluções de entrada.</i> .....	272
<b>Tabela 7.6</b> – <i>Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.</i> .....	273
<b>Tabela 7.7</b> – <i>Desempenho das melhores redes de segmentação semântica por classe no conjunto de teste composto por blocos do ortomosaico RGB.</i> .....	279
<b>Tabela 7.8</b> – <i>Desempenho IoU médio dos no conjunto de validação e teste composto por blocos do mosaico multiespectral RedEdge-Green-NIR.</i> .....	285
<b>Tabela 7.9</b> – <i>Desempenho das redes de segmentação semântica por classe no conjunto de teste composto por blocos válidos do mosaico multiespectral.</i> .....	285
<b>Tabela A.1</b> – <i>Imagens originais selecionadas para treinamento/validação e teste.</i> .....	309

# Sumário

<b>1 INTRODUÇÃO .....</b>	<b>1</b>
1.1 Motivação .....	1
1.2 Definição do problema .....	4
1.3 Objetivos .....	5
1.4 Contribuições .....	6
1.5 Organização do trabalho .....	7
<b>2 FUNDAMENTOS DE REDES NEURAS .....</b>	<b>9</b>
2.1 Redes neurais artificiais .....	12
2.1.1 <i>Gradiente descendente com backpropagation</i> .....	16
2.1.2 <i>Generalização e sobreajuste</i> .....	17
2.2 Redes neurais convolucionais .....	20
2.2.1 <i>Arquitetura de rede neural convolucional</i> .....	21
2.2.1.1 <i>Camada convolucional</i> .....	22
2.2.1.2 <i>Camada de subamostragem</i> .....	25
2.2.1.3 <i>Camadas totalmente conectadas</i> .....	27
2.3 Arquiteturas de classificação de imagens .....	28
2.3.1 <i>Modelo AlexNet</i> .....	28
2.3.2 <i>Modelo VGG</i> .....	32
2.3.3 <i>Modelos Inceptions</i> .....	35
2.3.4 <i>Modelo ResNet</i> .....	39
2.3.5 <i>Modelo Xception</i> .....	43
2.3.6 <i>Modelo DenseNet</i> .....	47
2.3.7 <i>Comparações entre redes</i> .....	50
2.4 Arquiteturas de segmentação semântica .....	52
2.4.1 <i>Modelo Hypercolumns</i> .....	56
2.4.2 <i>Modelo FCN</i> .....	57
2.4.3 <i>Modelo U-Net</i> .....	64
2.4.4 <i>Modelo DeconvNet</i> .....	66
2.4.5 <i>Modelo SegNet</i> .....	69
2.4.6 <i>Modelo RefineNet</i> .....	72
2.4.7 <i>Modelo DilatedNet</i> .....	74
2.4.8 <i>Modelos DeepLab</i> .....	79
2.4.8.1 <i>Modelo DeepLab-v2</i> .....	81
2.4.8.2 <i>Modelo DeepLab-v3</i> .....	84
2.4.8.3 <i>Modelo DeepLab-v3+</i> .....	88
2.4.9 <i>Modelo DenseASPP</i> .....	91
2.4.10 <i>Modelo FC-DenseNet</i> .....	95
2.4.11 <i>Modelo BiSeNet</i> .....	97
2.4.12 <i>Comparações entre redes</i> .....	100
2.4.13 <i>Desafios da segmentação semântica</i> .....	103

2.5	Transferência de aprendizado .....	108
2.5.1	<i>Estratégias de transferência de aprendizado</i> .....	108
2.5.2	<i>Transferência de aprendizado da CNN para redes de segmentação</i> .....	109
2.5.3	<i>Ajuste fino</i> .....	112
<b>3</b>	<b>TRABALHOS CORRELATOS DE SEGMENTAÇÃO SEMÂNTICA NA AGRICULTURA .....</b>	<b>115</b>
3.1	Imagens multiespectrais e mapas de índices de vegetação .....	117
3.2	<i>Weed detection in soybean crops using convnets</i> .....	122
3.3	<i>WeedNet: Dense semantic weed classification using multispectral images and MAV for smart farming</i> .....	124
3.3.1	<i>Aquisição do conjunto de dados</i> .....	124
3.3.2	<i>Arquitetura de segmentação semântica densa</i> .....	126
3.3.3	<i>Resultados quantitativos</i> .....	126
3.3.4	<i>Resultados qualitativos</i> .....	127
3.4	<i>WeedMap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming</i> .....	128
3.4.1	<i>Aquisição do conjunto de dados</i> .....	128
3.4.2	<i>Arquitetura de segmentação semântica densa</i> .....	131
3.4.3	<i>Resultados quantitativos</i> .....	132
3.4.4	<i>Resultados qualitativos</i> .....	133
3.5	Análise e direcionamentos .....	135
<b>4</b>	<b>AQUISIÇÃO DO CONJUNTO DE DADOS .....</b>	<b>137</b>
4.1	Fluxo de trabalho .....	137
4.2	Aquisição de imagens aéreas com VANT .....	138
4.2.1	<i>Especificação da plataforma, câmera e sensores</i> .....	139
4.2.2	<i>Resolução espacial das imagens aéreas</i> .....	140
4.2.3	<i>Conjunto de imagens aéreas</i> .....	141
4.2.4	<i>Conjunto de pontos de controle georreferenciados</i> .....	145
4.3	Etapas de processamento de imagens aéreas .....	145
4.3.1	<i>Etapa 1 – Processamento inicial</i> .....	147
4.3.2	<i>Etapa 2 – Nuvem de pontos densificada e malha 3D</i> .....	150
4.3.3	<i>Etapa 3 – Modelos digitais de elevação</i> .....	151
4.3.4	<i>Ortorretificação de imagens aéreas (ortofotos)</i> .....	152
4.3.4.1	<i>Distorções da lente</i> .....	152
4.3.4.2	<i>Distorções por obturador eletrônico</i> .....	153
4.3.4.3	<i>Distorções de inclinação da câmera</i> .....	154
4.3.4.4	<i>Distorções de perspectiva cônica</i> .....	155
4.3.4.5	<i>Distorções de relevo</i> .....	157
4.3.4.6	<i>Ortorretificação de imagens do mosaico</i> .....	158
4.3.5	<i>Etapa 4 – Ortomosaico 2D</i> .....	158
4.3.6	<i>Etapa 4 – Mapas de reflectância</i> .....	160
4.3.6.1	<i>Calibração radiométrica de imagens multiespectrais</i> .....	160
4.3.7	<i>Etapa adicional: Mapas de índices de vegetação</i> .....	162
4.4	Processamento de imagens do campo experimental .....	164
4.4.1	<i>Processamento inicial e nuvem de pontos densificada</i> .....	164
4.4.2	<i>Modelo digital de superfície RGB e multiespectral</i> .....	166
4.4.3	<i>Ortomosaico RGB e mapas de reflectância multiespectrais</i> .....	166
4.4.4	<i>Mapas de índice de vegetação</i> .....	170
4.4.5	<i>Imagem multicanal e composição colorida</i> .....	173
4.4.5.1	<i>Desalinhamento entre imagens originais multiespectrais</i> .....	175
4.4.5.2	<i>Desalinhamento entre imagens originais RGB e multiespectrais</i> .....	176
4.4.5.3	<i>Desalinhamento entre mosaico RGB e mapas de reflectância</i> .....	176

<b>5</b>	<b>SEGMENTAÇÃO SEMÂNTICA DE IMAGENS AÉREAS .....</b>	<b>179</b>
5.1	Preparação do conjunto de dados .....	181
5.1.1	<i>Rotulação das imagens originais</i> .....	181
5.1.2	<i>Blocos de imagens originais RGB e groundtruths</i> .....	183
5.1.3	<i>Rotulação do ortomosaico RGB</i> .....	185
5.1.4	<i>Blocos do ortomosaico RGB e groundtruth</i> .....	187
5.1.5	<i>Rotulação de mosaico multiespectral multicanal</i> .....	189
5.2	Experimentos com redes de segmentação semântica .....	194
5.2.1	<i>Implementação das redes de segmentação semântica</i> .....	194
5.2.2	<i>Estratégias de treinamento</i> .....	195
5.2.3	<i>Treinamento/validação e teste das redes de segmentação</i> .....	196
5.2.3.1	<i>Treinamento/validação com blocos de imagens RGB</i> .....	197
5.2.3.2	<i>Teste com blocos de imagens RGB</i> .....	199
5.2.3.3	<i>Predição com blocos de teste das imagens RGB</i> .....	200
5.2.3.4	<i>Avaliação de desempenho com blocos de teste das imagens RGB</i> .....	201
5.2.3.5	<i>Predição com blocos do mosaico RGB</i> .....	202
5.2.3.6	<i>Avaliação de desempenho com blocos do mosaico RGB</i> .....	203
5.2.3.7	<i>Composição dos blocos preditos do mosaico RGB</i> .....	204
5.3	Resultados das redes de segmentação semântica .....	205
5.3.1	<i>Resultados quantitativos com imagens aéreas RGB</i> .....	205
5.3.1.1	<i>Medidas de desempenho de treinamento e validação</i> .....	205
5.3.1.2	<i>Medidas de desempenho de teste</i> .....	210
5.3.1.3	<i>Testes adicionais em diferentes resoluções</i> .....	213
5.3.1.4	<i>Testes adicionais com diferentes hiperparâmetros de treinamento</i> .....	217
5.3.2	<i>Resultados qualitativos com imagens de teste RGB</i> .....	222
5.3.2.1	<i>Teste com tamanhos de blocos e GSD diferentes do treinamento</i> .....	226
5.3.2.2	<i>Treinamento com diferentes tamanhos de blocos e GSD</i> .....	228
5.3.3	<i>Resultados quantitativos com mosaico de teste RGB</i> .....	231
5.3.4	<i>Resultados qualitativos com mosaico de teste RGB</i> .....	232
5.3.4.1	<i>BiSeNet com diferentes tamanhos de bloco e GSD de treinamento</i> .....	237
5.3.4.2	<i>BiSeNet com diferentes tamanhos de batch</i> .....	238
5.3.4.3	<i>BiSeNet (2 x GSD) testado com diferentes tamanhos e GSD</i> .....	239
5.3.4.4	<i>BiSeNet (1 x GSD) testado com diferentes tamanhos e GSD</i> .....	241
5.3.5	<i>Análise de requisitos para segmentação semântica na agricultura</i> .....	243
<b>6</b>	<b>MODELOS DE SEGMENTAÇÃO MULTIESCALA .....</b>	<b>245</b>
6.1	Introdução .....	245
6.2	Descrição dos modelos .....	247
6.2.1	<i>Modelo de referência</i> .....	247
6.2.2	<i>Modelo com bloco de convolução denso paralelo</i> .....	250
6.2.3	<i>Modelo com bloco denso paralelo e concatenação de atributos</i> .....	251
6.2.4	<i>Modelo com bloco denso paralelo e maior contexto</i> .....	253
6.2.5	<i>Modelos sem efeito multiescala</i> .....	255
6.2.5.1	<i>Bloco denso paralelo sem mapas intermediários</i> .....	255
6.2.5.2	<i>Bloco denso inverso</i> .....	256
6.3	Resultados quantitativos dos modelos desenvolvidos .....	258
6.3.1	<i>Resultados quantitativos com imagens aéreas RGB</i> .....	258
6.3.1.1	<i>Medidas de desempenho de treinamento e validação</i> .....	259
6.3.1.2	<i>Medidas de desempenho de teste</i> .....	261
6.4	Modelo proposto .....	263

<b>7 ANÁLISE E COMPARAÇÃO DE RESULTADOS .....</b>	<b>269</b>
7.1 Resultados quantitativos com imagens aéreas RGB.....	269
7.1.1 <i>Medidas de desempenho de validação e teste</i> .....	270
7.1.2 <i>Resultados qualitativos com imagens de teste RGB</i> .....	274
7.2 Resultados quantitativos com mosaico de teste RGB .....	279
7.2.1 <i>Resultados qualitativos com mosaico de teste RGB</i> .....	280
7.3 Resultados com mosaico multiespectral multicanal .....	285
<b>8 AVALIAÇÃO EM CONJUNTO DE DADOS EXTRA.....</b>	<b>289</b>
8.1 Mosaicos RGB e <i>groundtruths</i> .....	289
8.2 Treinamento com blocos de $512 \times 512$ dos mosaicos RGB .....	292
8.2.1 <i>Resultado qualitativo de teste com bloco de <math>512 \times 512</math> (1 x GSD)</i> .....	294
8.2.2 <i>Resultado qualitativo de teste com bloco de <math>512 \times 256</math> (2 x GSD)</i> .....	296
8.3 Treinamento com blocos de $256 \times 256$ dos mosaicos RGB .....	298
8.3.1 <i>Resultado qualitativo de teste de <math>256 \times 256</math> (1 x GSD)</i> .....	299
<b>9 CONCLUSÕES .....</b>	<b>301</b>
9.1 Discussão dos resultados.....	301
9.1.1 <i>Redes de segmentação semântica</i> .....	302
9.1.2 <i>Modelo de rede proposta</i> .....	305
9.2 Limitações e trabalhos futuros .....	305
<b>A CONJUNTOS DE DADOS .....</b>	<b>307</b>
A.1 Imagens RGB originais calibradas.....	307
A.2 Imagens RGB selecionadas e <i>groundtruths</i> .....	309
A.3 Mosaico RGB e <i>groundtruth</i> .....	316
A.4 Composições multiespectrais e <i>groundtruth</i> .....	319
<b>B MODELOS DE SEGMENTAÇÃO SEMÂNTICA .....</b>	<b>323</b>
B.1 Modelo FCN .....	325
B.2 Modelo UNet.....	326
B.3 Modelo SegNet.....	327
B.4 Modelo RefineNet .....	327
B.5 Modelo BiSegNet .....	329
B.6 Modelo PSPNet.....	330
B.7 Modelo DeepLabV3 .....	331
B.8 Modelo DeepLabV3Plus .....	332
B.9 Modelo DenseASPP .....	334
<b>C MÉTRICAS DE DESEMPENHO .....</b>	<b>337</b>
C.1 Matriz de confusão .....	337
C.2 Intersecção sobre união para detecção de objetos .....	340
C.3 Intersecção sobre união para segmentação de imagens.....	342
<b>D INFORMAÇÕES ADICIONAIS .....</b>	<b>343</b>
D.1 Função de ativação Softmax.....	343
D.2 Convolução transposta.....	344
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>345</b>



# Capítulo 1

## Introdução

Neste capítulo descrevemos a motivação para desenvolvimento deste trabalho que explora técnicas avançadas de aprendizado de máquina da área de visão computacional. Como tema principal desta tese, abordamos a aplicação de redes de segmentação semântica baseadas em redes neurais convolucionais profundas (CNN), particularmente em mosaico de imagens de alta resolução de um veículo aéreo não tripulado (VANT), para mapeamento da distribuição de plantas daninhas na cultura de cana-de-açúcar.

### 1.1 Motivação

Biocombustíveis são fontes renováveis de energia que podem substituir, parcial ou totalmente, combustíveis derivados de petróleo e gás natural. A produção de biocombustíveis de forma sustentável, com destaque para o etanol derivado da cana-de-açúcar, será estratégica ao cumprimento das metas de redução das emissões de CO<sub>2</sub>, com o objetivo de minimizar as consequências do aquecimento global. Com a instituição da Política Nacional de Biocombustíveis (RenovaBio), o Brasil espera reduzir as emissões de gases do efeito estufa, conforme assinado no Acordo de Paris [RENOVABIO]. No entanto, para lidar com a crescente demanda por produtos derivados de cana-de-açúcar, muitos agricultores expandem as plantações de cana para novas áreas, desmatando florestas e deslocando culturas tradicionais, com a consequente perda de biodiversidade e impacto ambiental. Portanto, aumentar o rendimento na agricultura é fundamental para reduzir os danos ambientais causados pela expansão agrícola [LUNA e LOBO 2016].

Um dos problemas que diminui o rendimento da lavoura é a infestação de plantas daninhas<sup>1</sup> no campo. O manejo inadequado e aplicação uniforme de herbicidas aumenta a resistência das plantas daninhas. Isso compromete o desenvolvimento de diversas culturas agrícolas, particularmente dos canaviais, provocando sérias perdas na produtividade, aumento dos custos de produção, dificuldades na colheita, depreciação da qualidade da matéria-prima na indústria sucroalcooleira e, conseqüentemente, diminuição do valor comercial das áreas cultivadas [BRIGHENTI 2010]. Desta forma, há necessidade de um manejo adequado da área de cultivo, com o uso de ferramentas de agricultura de precisão (AP) que possibilitem aos agricultores um maior controle sobre a produção agrícola. A agricultura de precisão utiliza tecnologias avançadas para coletar dados geograficamente referenciados, permitindo uma análise detalhada da área de cultivo – como falhas no plantio, matocompetição e doenças das plantas – proporcionando uma tomada de decisão

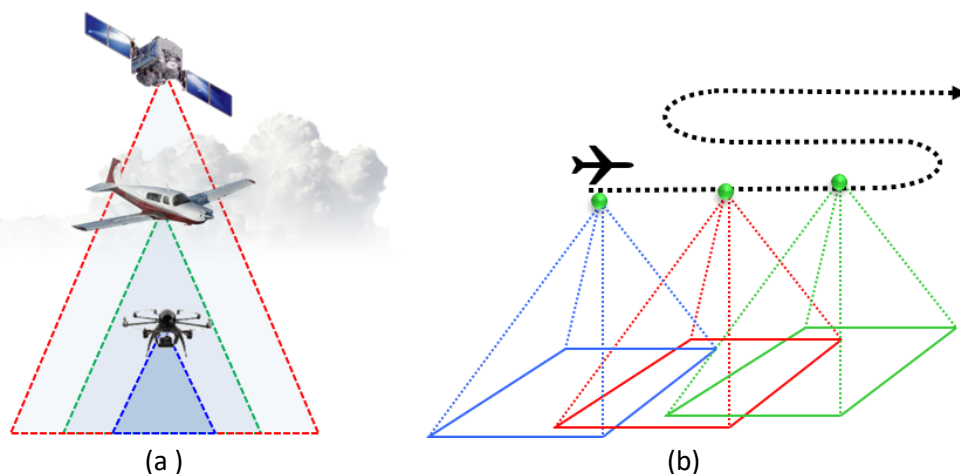
---

<sup>1</sup> Plantas daninhas são plantas invasoras e indesejáveis, ou outras plantas que, mesmo sendo úteis, nascem no meio de outra cultura, competindo por água, luz, nutrientes e espaço, causando danos na plantação.

mais assertiva [INAMASU *et al.* 2011]. Um dos objetivos da AP é permitir a aplicação dos insumos em taxas variáveis nas doses certas, no momento certo e no lugar certo, como por exemplo, aplicação localizada de herbicidas para controle de plantas invasoras, aumentando a produtividade e minimizando os custos de produção, com efeitos na redução do impacto ambiental, como um passo relevante em direção a uma agricultura sustentável [FILHO e CHRISTOFFOLETI 1987].

O sensoriamento remoto por satélite é uma das ferramentas da agricultura de precisão utilizada para mapeamento aéreo de grandes áreas de cultura de cana-de-açúcar, com objetivo de detecção de estresse (causado por doenças, pragas ou deficiências de nutrientes e água) e previsão de rendimento da produção [LUNA e LOBO 2016]. Porém, o mapeamento aéreo por imagens de satélites ou aeronaves tripuladas, embora tenha a vantagem de cobrir grandes áreas, apresenta baixa resolução espacial devido à longa distância entre a câmera e o terreno fotografado, causando dificuldades na discriminação das plantas daninhas [BRYSON *et al.* 2010]. Além disso, as imagens de satélites orbitais apresentam baixa resolução temporal devido à baixa frequência de passagem do sensor sobre a mesma região, podendo levar dias ou semanas para revisitar a área de interesse. A possibilidade de ocorrência de nuvens também impede a visualização sistemática e periódica do terreno nos momentos críticos de desenvolvimento da cultura [TORRES-SÁNCHEZ *et al.* 2013]. Como consequência da baixa resolução temporal, os atrasos no fornecimento de informações ao agricultor e, conseqüentemente, na tomada de decisão, podem interferir no desempenho das práticas de manejo para controle das plantas daninhas em tempo real durante a safra.

Como estratégia complementar ao sensoriamento remoto por satélite (Figura 1.1a), uma tecnologia mais recente tem conquistado cada vez mais espaço na agricultura de precisão com o uso de veículos aéreos não tripulados (VANT), ou, em inglês, UAV (*Unmanned Aerial Vehicle*), popularmente conhecidos como drones [HUNG *et al.* 2014; JORGE e INAMASU 2014; SHI *et al.* 2016]. Ao realizar um mapeamento aéreo, o VANT registra imagens sobrepostas de alta resolução espacial da região de interesse, que posteriormente são agrupadas em um único mapa ortomosaico georreferenciado de alta resolução, cobrindo áreas (talhões) agrícolas de pequeno/médio porte (Figura 1.1b) [SUZUKI *et al.* 2010]. Com estes mapas georreferenciados é possível obter informações sobre a localização, distribuição e variabilidade das plantas invasoras no campo.

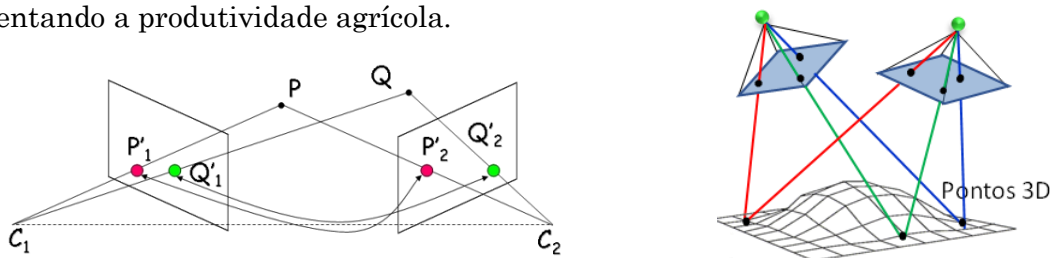


**Figura 1.1** – Sensoriamento remoto com diferentes áreas de cobertura do solo (a), dependendo da altura da plataforma (satélite, aeronave ou veículo aéreo não tripulado); imagens sobrepostas ao longo de uma trajetória de voo sobre a região de interesse (b).

Uma das vantagens dos VANTs em relação aos satélites é a aquisição de imagens de alta resolução espacial, da ordem de poucos centímetros por *pixel*, devido à menor altura de voo e proximidade com o terreno (Figura 1.1). A resolução espacial depende da altura de voo acima do solo e das características da câmera, especificadas de acordo com os requisitos da aplicação, como por exemplo, necessidade de identificar espécies individuais de plantas daninhas, estágio de desenvolvimento das plantas, tamanho da área, entre outros. Porém, voos muito baixos são impraticáveis devido à pequena cobertura de área fotografada no terreno e à limitada autonomia de voo dos VANTs (ou seja, baixa duração da bateria) durante o mapeamento de grandes extensões de terra. Dentre outras vantagens dos VANTs em relação às plataformas de satélites e aeronaves, destacam-se: controle automático do posicionamento da câmera no plano de voo; boa resolução temporal para monitoramento periódico das culturas, revisitando os locais de interesse sempre que necessário; e sobrevoo abaixo da linha das nuvens (< 120 metros), sem perda de qualidade nas imagens ou oclusão em dias nublados [PEÑA *et al.* 2015; TORRES-SÁNCHEZ *et al.* 2013].

Embora os sensores óticos a bordo de VANTs forneçam imagens de alta resolução espacial, cada imagem cobre apenas uma pequena área de interesse. Assim, após o planejamento da trajetória de voo para captura automática de imagens com sobreposição no solo, softwares aerofotogramétricos realizam um pós-processamento das imagens aéreas para gerar um mapa ortomosaico de alta resolução, com maior cobertura da área total de interesse. Os softwares aerofotogramétricos são bastante utilizados há décadas para processamento de imagens aéreas capturadas por aviões equipados com câmeras fotogramétricas de alto custo e precisão [WOLF *et al.* 2014]. Mais recentemente, surgiram alternativas de softwares comerciais e de código aberto, adaptados para imagens de VANTs, já que estes utilizam câmeras não fotogramétricas e sensores GPS de baixo custo e precisão [MATTIVI *et al.* 2021]. Estes softwares modernos usam uma técnica de visão computacional, conhecida como *Structure from Motion* (SfM) [HARTLEY e ZISSERMAN 2003; WESTOBY *et al.* 2012], baseada em métodos tradicionais de fotogrametria usando estereoscopia, para gerar uma reconstrução 3D do terreno (Figura 1.2) e outros produtos derivados, como um mosaico ortorretificado a partir das imagens aéreas sobrepostas.

Com os mapas ortomosaicos e outros produtos gerados, diversos métodos de análise usando sistemas de informação geográfica (SIG) são capazes de orientar os produtores rurais a tomar decisões sobre a saúde da vegetação, qualidade do solo, nutrição, pragas, stress hídrico, etc. Como os mapas são georreferenciados, é possível localizar os pontos críticos da plantação com ocorrências de plantas daninhas para tomar ações pontuais de manejo. Deste modo, os mapas da distribuição de plantas daninhas fornecem informações importantes para gerar mapas de prescrição de insumos no campo de cana-de-açúcar que, por conseguinte, podem ser transferidos para máquinas automatizadas equipadas com GPS, como pulverizadores de herbicidas ou fertilizantes. Assim, este procedimento permite minimizar o uso de produtos químicos, reduzindo os impactos ambientais e aumentando a produtividade agrícola.



**Figura 1.2** – Estereoscopia<sup>2</sup> para reconstruir um objeto 3D (terreno) imageado por pelo menos dois pontos de vista diferentes (imagens estereoscópicas da cena) [JENSEN 2009].

<sup>2</sup> A profundidade dos pontos P e Q do objeto real visualizadas pelas câmeras C1 e C2 em posição conhecida podem ser recuperadas usando triangulação, usando os pares de pontos 1 e 2 na primeira e segunda imagem.

## 1.2 Definição do problema

Na literatura, já existem diversos métodos de visão computacional para detectar plantas invasoras a partir de imagens de sensoriamento remoto por satélite, incluindo os métodos tradicionais de aprendizado de máquina, como: *K-Nearest Neighbor* (KNN) [COVER *et al.* 1967], máquinas de vetores de suporte (SVM) [CORTES e VAPNIK 1995; SUYKENS *et al.* 1999], entre outros. Entretanto, apesar dos importantes avanços nos sistemas de aquisição de VANTs, a detecção automática de ervas daninhas continua sendo um problema desafiador. Com o aumento da resolução espacial nas imagens de VANTs e a maior flexibilidade na escolha de sensores óticos – RGB e multiespectrais – outras técnicas mais modernas de inteligência artificial usando redes neurais vêm sendo desenvolvidas nos últimos anos para abordar vários problemas, inclusive detecção de plantas daninhas. Além disso, com as imagens multiespectrais disponíveis, é possível obter mapas de índices de vegetação que fornecem informações importantes a respeito das condições de determinada cultura agrícola, como por exemplo, saúde da planta e atividade fotossintética a partir da refletância de cada estrutura foliar [HOLLER *et al.* 2022].

Os algoritmos de aprendizado profundo (*Deep Learning* – DL) constituem uma inovação na área de aprendizado de máquina que vem sendo usada com sucesso em várias áreas, com grande potencial de aplicação [LECUN *et al.* 2015]. Mais recentemente, as redes neurais convolucionais (*Convolutional Neural Network* – CNN), que integram uma classe específica de DL, vêm sendo aplicadas a diversos desafios de visão computacional relacionados à produção agrícola [KAMILARIS e PRENAFETA-BOLDÚ 2018]. Pesquisas recentes compararam as vantagens e desvantagens do uso de CNNs em relação a outras técnicas tradicionais de aprendizado de máquina aplicadas na agricultura. Os resultados gerais indicam que a CNN é uma técnica promissora com alto desempenho em termos de precisão de classificação, superando as técnicas tradicionais.

Além da classificação de imagens, a segmentação semântica, que atribui rótulos semânticos a cada *pixel* de uma imagem, é um dos tópicos fundamentais em visão computacional. As CNNs de classificação de imagens vêm sendo usadas com sucesso em técnicas de segmentação semântica em diversas áreas, como robótica, com a finalidade de entendimento de cenas externas (ruas, prédios, carros, etc.) ou internas (mesa, cadeira, etc.) para navegação autônoma. Recentemente, as redes de segmentação semântica baseadas em CNN vêm sendo utilizadas inclusive em aplicações na agricultura [HUANG *et al.* 2018; SA *et al.* 2017; SA *et al.* 2018]. No entanto, o desempenho de cada rede de segmentação semântica é dependente da qualidade e tamanho do conjunto de dados rotulado, que serve como entrada para treinamento da rede [KAMILARIS e PRENAFETA-BOLDÚ 2018]. Porém, não existem muitos conjuntos de dados no domínio da agricultura disponíveis publicamente [SA *et al.* 2017].

Na maioria das pesquisas, o conjunto de dados usado para treinar a rede de classificação/segmentação consiste de um pequeno conjunto de dados de imagens capturadas por satélite ou VANTs. Em geral, quanto mais complexo o problema a ser resolvido (por exemplo, maior número de classes a serem previstas ou diferentes escalas dos objetos), mais dados são necessários para aumentar a acurácia da classificação. Entretanto, a rotulação de grandes conjuntos de dados agrícolas por um especialista, com anotações no nível de *pixel*, é uma tarefa demorada e, em muitos casos, os próprios pesquisadores precisam produzir seus conjuntos de imagens rotuladas manualmente.

Neste trabalho, nos concentramos em modelos de redes de segmentação semântica por serem mais adequados para identificação de plantas daninhas do que os modelos de classificação de imagens, detecção de objetos ou segmentação de instâncias, onde os limites dos objetos em ambientes agrícolas não são bem-definidos. Portanto, esta tese tem como tema principal a aplicação de algumas arquiteturas diferentes de redes de segmentação semântica baseadas em redes neurais convolucionais para mapeamento da distribuição de plantas daninhas em ortomosaicos georreferenciados de imagens aéreas de uma cultura de cana-de-açúcar capturadas por VANT, como ferramenta de auxílio na área de agricultura de precisão.

Realizamos uma avaliação de desempenho quantitativo e qualitativo de diversas redes existentes de segmentação semântica baseadas em CNN, aplicadas em nosso conjunto de dados de domínio específico com quatro classes semânticas (solo, cultura, daninhas e gramínea). Com base na análise dos resultados obtidos e características destas redes, propomos uma nova rede de segmentação semântica multiescala com melhor desempenho do que as redes avaliadas. A rede proposta foi projetada especificamente para detectar daninhas de diferentes tamanhos em mosaicos de alta resolução com escalas variáveis (devido a variações da altura de voo ou da elevação do terreno durante o mapeamento aéreo) ou em mosaicos com escalas diferentes da originalmente treinada pela rede, sem necessidade de estratégias de treinamento com imagens de entrada em diferentes resoluções espaciais.

### 1.3 Objetivos

Nossa aplicação de segmentação semântica, visando o mapeamento da distribuição de plantas daninhas em um mosaico de imagens aéreas, consiste na tarefa de classificação de cada *pixel* em quatro classes (solo, cultura, erva daninha de folhas largas e gramíneas de folhas estreitas). Para isso, primeiro selecionamos algumas redes recentes de segmentação baseadas em CNN, com *backbones* pré-treinados em conjuntos de dados de domínio geral, como ImageNet [DENG *et al.* 2009]; depois, ajustamos os pesos da rede em nosso conjunto de dados específico; e, por fim, avaliamos os resultados de acordo com uma métrica de desempenho. Os modelos de rede FCN [LONG *et al.* 2015], U-Net [RONNEBERGER *et al.* 2015], SegNet [BADRINARAYANAN *et al.* 2017], RefineNet [LIN, MILAN *et al.* 2017], PSPNet [ZHAO *et al.* 2017], DeepLab-v3 [CHEN *et al.* 2017], DeepLab-v3+ [CHEN *et al.* 2018b], DenseASPP [YANG *et al.* 2018] e BiSeNet [YU *et al.* 2018] foram selecionados devido ao potencial de extração de atributos a nível de *pixel* e aos bons resultados em diferentes conjuntos de dados de referência de domínio geral na tarefa de segmentação semântica de imagens naturais, como PASCAL-VOC [EVERINGHAM *et al.* 2015].

Um dos problemas das redes de segmentação semântica treinadas em outros conjuntos de dados, quando aplicadas no domínio da agricultura, é o baixo desempenho de segmentação, em particular, das ervas daninhas, devido a diferenças visuais entre os conjuntos de dados dos domínios geral e específico, bem como uma quantidade limitada de instâncias de plantas invasoras no campo e suas semelhanças espectrais com a cultura. Além disso, imagens aéreas de VANT podem ter diferentes resoluções espaciais dependendo da altura de voo. Com isso, os métodos de segmentação devem ter a capacidade de reconhecer objetos multiescala para obter melhores resultados.

Desta forma, o objetivo principal deste trabalho é melhorar o desempenho das redes de segmentação semântica em mosaicos aéreos de uma cultura de cana-de-açúcar. Para isso, analisamos as características de algumas redes de segmentação semântica existentes e desenvolvemos uma nova arquitetura multiescala baseada em CNN, treinada em nosso conjunto de dados experimental. Por fim, avaliamos e comparamos nossos resultados com os das redes avaliadas, usando uma métrica qualitativa de desempenho e uma análise visual dos mapas de predição semântica.

## 1.4 Contribuições

Para desenvolvimento do trabalho apresentado nesta tese, realizamos as seguintes atividades de pesquisa:

- análise das características das imagens aéreas RGB e multiespectrais de VANT, bem como do processo de geração dos seus respectivos ortomosaicos georreferenciados, que impactam no desempenho das redes de segmentação semântica na área agrícola;
- análise de diferentes arquiteturas de redes de segmentação semântica baseadas em redes neurais convolucionais, examinando aspectos importantes para aplicações de agricultura de precisão usando mosaicos de alta resolução espacial;
- avaliação de desempenho de nove redes de segmentação semântica existentes, pré-treinadas e ajustadas em nosso domínio específico, para mapeamento da distribuição de plantas daninhas na cultura de cana-de-açúcar;
- desenvolvimento de uma nova arquitetura de rede de segmentação semântica baseada em rede neural convolucional, que agrega informações de contexto multiescala, projetada especificamente para aplicações agrícolas;
- avaliação do desempenho da rede proposta em um conjunto de dados RGB e multiespectral para comparação com as redes existentes avaliadas neste trabalho.

Com isso, as principais contribuições obtidas neste trabalho foram:

- rotulação e criação de bases de dados a partir de imagens aéreas capturadas por VANT e mosaicos RGB ou multiespectrais de alta resolução, úteis para treinamento das redes neurais de segmentação semântica por outros pesquisadores;
- rede proposta de segmentação semântica com desempenho superior às redes já existentes, especificamente quando aplicada ao problema de segmentação semântica de plantas daninhas em mosaico aéreo de resolução espacial variável;
- melhora na identificação das ervas daninhas na cultura de cana-de-açúcar.

Esperamos que este trabalho possa oferecer informações sobre os desafios e as limitações desta pesquisa, que contribuam para futuras pesquisas científicas relevantes na área de visão computacional e aprendizado profundo, por meio do desenvolvimento de novas arquiteturas e estratégias de treinamento de redes neurais de segmentação semântica direcionadas à área de agricultura de precisão.



## 1.5 Organização do trabalho

O restante desta tese está estruturado nos seguintes capítulos. O [Capítulo 2](#) aborda toda a fundamentação teórica sobre redes neurais de aprendizado profundo necessária para desenvolvimento deste trabalho. Nele, apresentamos a revisão da literatura de algumas redes de segmentação semântica de imagens e das redes neurais convolucionais de classificação usadas como modelo-base. No [Capítulo 3](#), são apresentados alguns trabalhos correlatos que abordam problemas similares de mapeamento de plantas daninhas em larga escala em diversas culturas, usando sensoriamento remoto por VANTs. O [Capítulo 4](#) descreve as características de uma plataforma de um veículo aéreo não tripulado e seus sensores óticos embarcados; o fluxo de trabalho para aquisição e processamento de imagens aéreas de VANT na agricultura; e, por fim, os procedimentos para geração do ortomosaico, mapas de reflectância e mapas de índice de vegetação georreferenciados. No final deste capítulo, apresentamos os mapas gerados pelo processamento de imagens aéreas de VANT de um campo experimental de cana-de-açúcar com presença de plantas daninhas para compor o conjunto de dados de nosso estudo. O [Capítulo 5](#) primeiro descreve os procedimentos para gerar nosso conjunto de dados de treinamento, validação e teste das redes de segmentação semântica em um domínio específico. Em seguida, avaliamos o desempenho quantitativo e qualitativo de nove redes de segmentação semântica pré-treinadas e ajustadas neste conjunto de dados. O [Capítulo 6](#) descreve o método proposto de segmentação semântica com contexto multiescala, baseado em redes neurais convolucionais. O [Capítulo 7](#) apresenta uma análise e comparação dos resultados experimentais do modelo proposto, em relação aos modelos melhor avaliados no [Capítulo 5](#). O [Capítulo 8](#) apresenta apenas uma análise qualitativa dos desempenhos de algumas redes em um conjunto de dados extra, formado por mosaicos RGB de menor resolução espacial devido à maior altura de voo. O [Capítulo 9](#) apresenta as discussões e conclusões deste trabalho, mostra os desafios e limitações desta pesquisa e apresenta direções para trabalhos futuros.

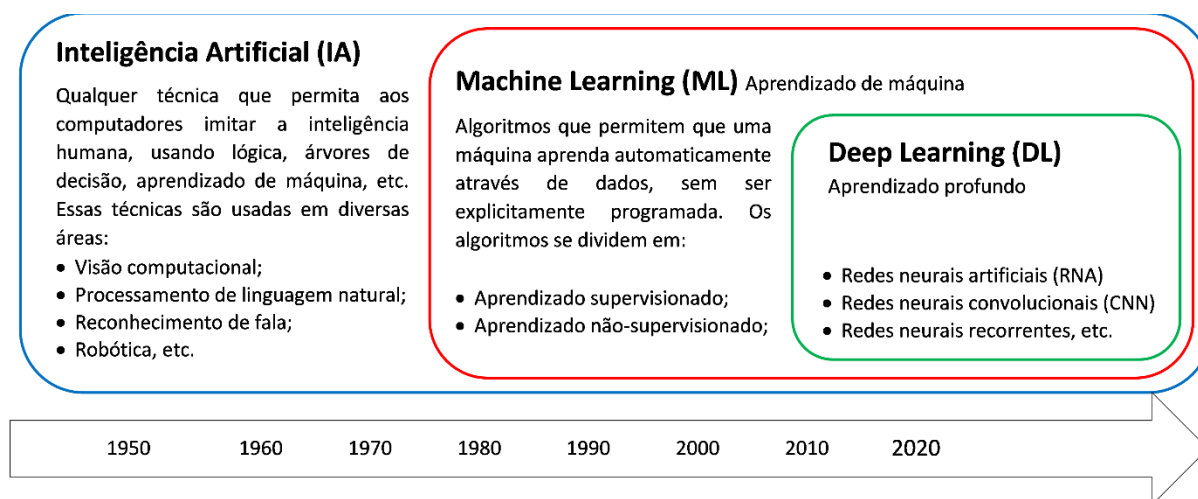




# Capítulo 2

## Fundamentos de redes neurais

Sistemas de visão computacional são baseados em técnicas de processamento de imagens que permitem extrair informações contidas nas imagens. Recentemente, muitas tarefas de visão computacional utilizam técnicas de inteligência artificial (IA). A área de IA abrange o aprendizado de máquina (*Machine Learning* – ML) e aprendizado profundo (*Deep Learning* – DL), como mostrado na [Figura 2.1](#). O aprendizado de máquina é um subcampo da inteligência artificial que desenvolve algoritmos com capacidade de aprendizado a partir de dados. O aprendizado profundo é um subcampo específico do aprendizado de máquina, onde modelos baseados em redes neurais profundas aprendem representações a partir de dados. Atualmente, as redes neurais de aprendizado profundo fornecem boas soluções para muitos problemas de reconhecimento de imagem, reconhecimento de fala, processamento de linguagem natural e robótica [[CHOLLET 2016](#)].

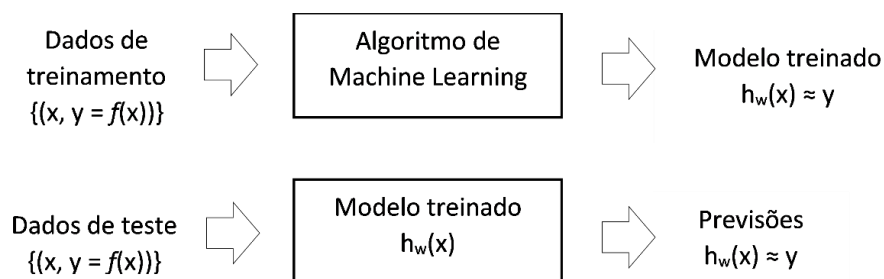


**Figura 2.1** – *Inteligência artificial, aprendizado de máquina e aprendizado profundo.*

Os modelos de aprendizado de máquina se dividem principalmente em aprendizado supervisionado e não supervisionado. Nesta tese, abordamos o aprendizado supervisionado que consiste em aprender a mapear dados de entrada para alvos de referência conhecidos (*labels* ou *groundtruths*), também chamados de anotações ou rótulos de classes de um domínio específico. As amostras de dados (exemplos ou instâncias) são denotadas por  $x$  e os alvos verdadeiros são denotados por  $y$ .

No aprendizado supervisionado, o modelo recebe um conjunto de entradas  $x$  com suas respectivas saídas  $y$  e tenta encontrar uma função que estabeleça uma relação aproximada entre elas. Mais formalmente, o modelo busca encontrar uma função  $h(x_i)$ , denominada hipótese, que se aproxime da função  $f(x_i)$ , para todos os dados de entrada  $x_i$ , onde  $f(x_i)$  é a saída da  $i$ -ésima entrada de  $x$ . Primeiro, o modelo deve ser treinado a partir de um conjunto de amostras de treinamento rotuladas  $(x, y)$ , como mostra a [Figura 2.2](#) superior. O objetivo é produzir um modelo com função paramétrica,  $h_W$ , definido pelos parâmetros (pesos) treinados,  $W$ , dentro de um espaço finito de hipóteses possíveis,  $H = \{h_{w_0}, h_{w_1}, \dots, h_{w_m}\}$ , que mapeia uma amostra de dados  $x$  para um valor previsto  $h_W(x)$ , que se aproxima da função alvo real  $y = f(x)$  desconhecida [[CHOLLET 2016](#)]. O conjunto de parâmetros  $W$ , que caracteriza o modelo treinado, é ajustado durante o treinamento de forma que minimize o erro de previsão, ou seja,  $h_W(x) \approx y$ .

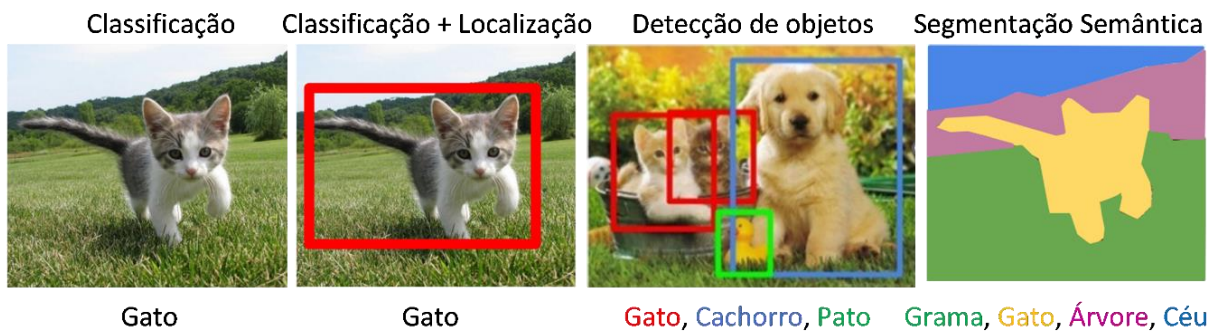
Como saída desse processo de treinamento, temos um modelo  $h_W$  treinado, em outras palavras, um modelo que aprendeu a relação dos dados  $(x, y)$ . Esse modelo treinado deve ser testado para realizar previsões sobre novos dados desconhecidos ([Figura 2.2](#) inferior). Esses dados de teste também estão rotulados, então basta comparar as previsões do modelo  $h_W(x)$ , com os respectivos rótulos  $y$  para analisar o desempenho das previsões. Se o modelo tem um bom desempenho de previsões sobre os dados de teste, espera-se que o modelo também tenha um bom desempenho para previsão de dados reais não rotulados.



**Figura 2.2** – Treinamento e teste de um modelo de aprendizado de máquina.

Uma tarefa comum de aprendizado supervisionado é a classificação de dados. Alguns algoritmos clássicos de aprendizado de máquina para classificação de dados são:  $K$ -vizinhos mais próximos ( $K$ -Nearest Neighbors – KNN) [[COVER et al. 1967](#)], máquinas de vetores de suporte (*Support Vector Machine* – SVM) [[CORTES e VAPNIK 1995](#); [SUYKENS et al. 1999](#)], árvores de decisão (*Decision Trees* – DT), floresta aleatória (*Random Forest* – RF) [[BREIMAN 2001](#)] e redes neurais.

Atualmente, as redes neurais de aprendizado profundo vêm sendo utilizadas com sucesso em diversas aplicações de visão computacional. Com a evolução dos sistemas paralelos de computação de alto desempenho com uso de GPUs (*Graphical Processing Units*); disponibilidade de grandes conjuntos de imagens para treinamento de redes neurais, como ImageNet [[DENG et al. 2009](#)]; e o desenvolvimento de algoritmos de aprendizado mais modernos, como o algoritmo de gradiente descendente estocástico (SGD), as redes neurais convolucionais (CNN) se tornaram a abordagem de aprendizado de máquina dominante para reconhecimento de objetos visuais [[CHOLLET 2016](#)] nas tarefas de classificação, localização e detecção de objetos, como também segmentação semântica de imagens, como ilustra a [Figura 2.3](#).



**Figura 2.3** – *Tarefas de visão computacional: classificação dos objetos; classificação e localização de caixa delimitadora de um objeto; detecção de vários objetos; e segmentação semântica densa a nível de pixel. Adaptado de [LI et al. 2017a].*

Na tarefa de classificação de imagens, cada imagem recebe um único rótulo (*groundtruth*) correspondente ao objeto principal na imagem. Assim, o conjunto de dados de treinamento consiste em imagens rotuladas com uma das categorias de objeto de um domínio específico. Após o treinamento, o sistema prevê a categoria do objeto presente em uma imagem do conjunto de teste e compara com o *groundtruth* [RUSSAKOVSKY et al. 2014].

A tarefa de localização retorna um rótulo de classe e uma caixa delimitadora (*bounding box*) de localização de um único objeto. O conjunto de dados de treinamento da tarefa de localização consiste em imagens rotuladas com um *groundtruth*, que indica uma das categorias de objeto, além de uma caixa delimitadora. Para cada imagem do conjunto de teste, o algoritmo prevê a categoria do objeto presente na imagem, além das coordenadas horizontal e vertical do centro do objeto, bem como a largura e altura da caixa delimitadora, indicando a posição e a escala do objeto [RUSSAKOVSKY et al. 2014].

A tarefa de detecção de objetos aborda o problema de classificação e localização de múltiplos objetos na imagem. O objetivo é especificar a localização de vários objetos na imagem, com seus respectivos rótulos de classe e caixas delimitadoras [SERMANET et al. 2013]. As imagens de treinamento são anotadas com caixas delimitadoras, indicando a posição e a escala de cada instância de cada categoria de objeto de destino. Para cada imagem de teste, os algoritmos preveem caixas delimitadoras, indicando a posição e a escala de todas as instâncias de todas as categorias de objetos de destino.

A segmentação semântica de imagens é uma tarefa que tem como objetivo classificar cada *pixel* de uma imagem em classes correspondentes de um domínio específico. Ao contrário da tarefa de classificação que atribui uma única classe à imagem inteira, a segmentação semântica classifica todos os *pixels* de forma densa, sem diferenciação de instâncias de objetos. Portanto, a saída de alta resolução geralmente tem o mesmo tamanho da imagem de entrada.

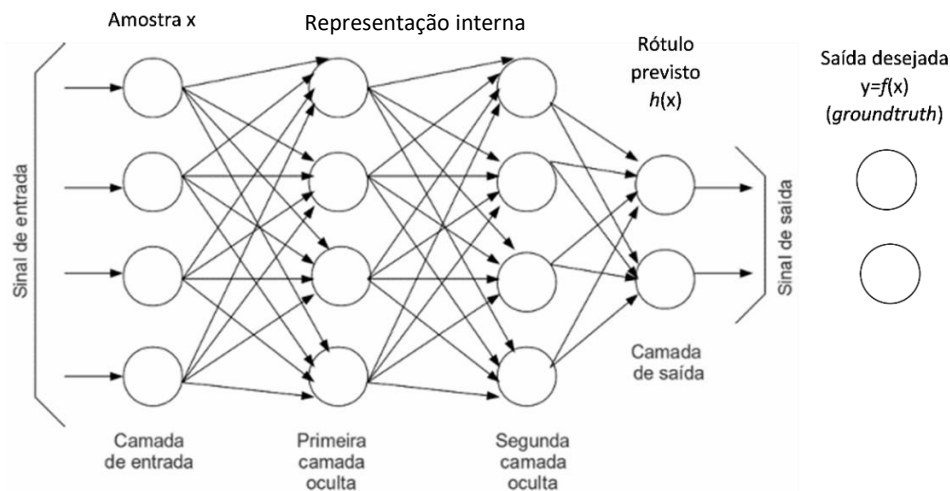
Neste trabalho, abordamos apenas os algoritmos de segmentação semântica de imagens baseados em CNNs de classificação de imagens. Portanto, o termo “detecção” será usado apenas como sinônimo de “localização” de plantas daninhas por meio de segmentação semântica densa a nível de *pixel*.

## 2.1 Redes neurais artificiais

As redes neurais artificiais são um paradigma de programação inspirado em neurônios biológicos do cérebro humano, no qual milhões de nós (neurônios matemáticos) de processamento são interconectados e organizados em camadas [GOODFELLOW *et al.* 2016]. A rede neural permite que um computador aprenda com os dados, ou seja, ela aprende a reconhecer automaticamente as classes de dados a partir de um conjunto de dados de treinamento rotulados [NIELSEN 2015]. Desta forma, muitas aplicações usam uma arquitetura de rede neural artificial tradicional (RNA), também chamada de rede totalmente conectada (*fully connected network*), que aprende a mapear uma entrada de tamanho fixo (por exemplo, um vetor com valores de *pixels* da imagem) para uma saída de tamanho fixo (por exemplo, uma probabilidade para cada uma das várias categorias ou classes), como mostrado na Figura 2.4.

Uma arquitetura de rede neural, também chamada de *MultiLayer Perceptron* – MLP [ROSENBLATT 1961], é composta de múltiplas camadas de neurônios (unidades), sendo: uma camada de entrada, com os neurônios de entrada; uma camada de saída, com neurônios de saída; e uma ou mais camadas ocultas, com neurônios intermediários, que não são nem entrada nem saída. Quando cada neurônio em uma camada está conectado a todos os neurônios na camada anterior, a camada é denominada totalmente conectada (*fully connected layer*) ou camada densa (*dense layer*). Uma rede neural de aprendizado profundo (*Deep Neural Network* – DNN) tem muitas camadas ocultas e pode processar grandes quantidades de dados [GÉRON 2019].

Nas redes *feedforward*, os sinais de entrada são propagados diretamente pelos neurônios, avançando das camadas de entrada até a camada de saída, passando pelas camadas ocultas intermediárias. Sendo assim, as redes podem aprender representações internas de dados (*features*, características ou atributos), onde as informações das unidades de entrada são reescritas em uma representação interna de baixo nível nas primeiras camadas, que por sua vez são reescritas em representações de maior nível de abstração nas camadas seguintes. Assim, as unidades de saída da rede são geradas a partir da representação interna de alto nível nas camadas mais profundas (e não pelo padrão original de entrada) [RUMELHART *et al.* 1986].

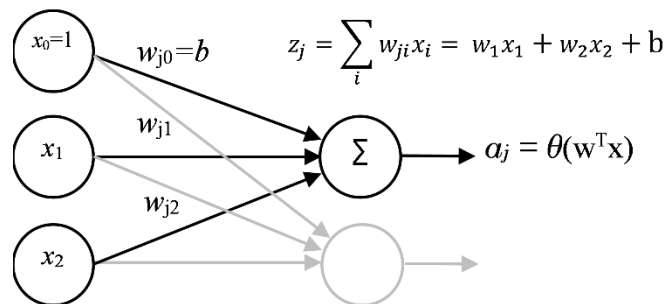


**Figura 2.4** – Arquitetura de uma rede neural multicamada totalmente conectada, composta por uma camada de entrada, duas camadas ocultas e uma camada de saída [LECUN 1989].

Uma amostra ou vetor de entrada  $x$  é apresentada à rede, definindo-se os estados das unidades de entrada  $x_i$ . Um neurônio artificial (unidade elementar da rede neural artificial) é uma função matemática inspirada em neurônios biológicos, onde o neurônio recebe um estímulo (entrada) e dispara uma resposta (saída ou ativação) de acordo com uma função de ativação [ROSENBLATT 1958]. Cada neurônio artificial realiza um produto escalar entre o vetor de entrada  $x$  e o vetor de peso  $w$  das conexões da camada anterior, ou seja, uma combinação linear ou soma das entradas  $x_i$  ponderadas pelos respectivos pesos  $w_i$ . Em outras palavras, em uma rede de  $L$  camadas, cada neurônio  $j$  em uma determinada camada  $l$  aplica um filtro linear às saídas dos neurônios da camada anterior  $l-1$  [SRIVASTAVA *et al.* 2014]. Assim, a somatória  $z_j$  na entrada da unidade  $j$  é uma função linear das suas entradas  $x_i$  que estão conectadas ao neurônio  $j$  e dos pesos de suas conexões  $w_{ji}$ , dada pela Equação 2.1. Normalmente, um viés escalar (*bias*) é adicionado à saída do filtro, mas para facilitar o cálculo, cada unidade recebe uma entrada extra (que sempre tem valor 1) e um peso  $b$  (viés). Desta forma, o viés pode ser tratado como os outros pesos na Equação 2.1, sendo o vetor coluna de peso  $w \in \mathbb{R}^{d+1} = \{[w_0, w_1, \dots, w_d]^T \mid w_0 = b \text{ e } w_i \in \mathbb{R}\}$  e o vetor coluna de entrada  $x \in \{1\} \times \mathbb{R}^d = \{[x_0, x_1, \dots, x_d]^T \mid x_0 = 1 \text{ e } x_i \in \mathbb{R}\}$ , onde  $d$  o número de unidades da camada anterior [ABU-MOSTAFA *et al.* 2012].

$$z_j = \sum_{i=0}^d w_{ji} x_i = w^T x \quad (2.1)$$

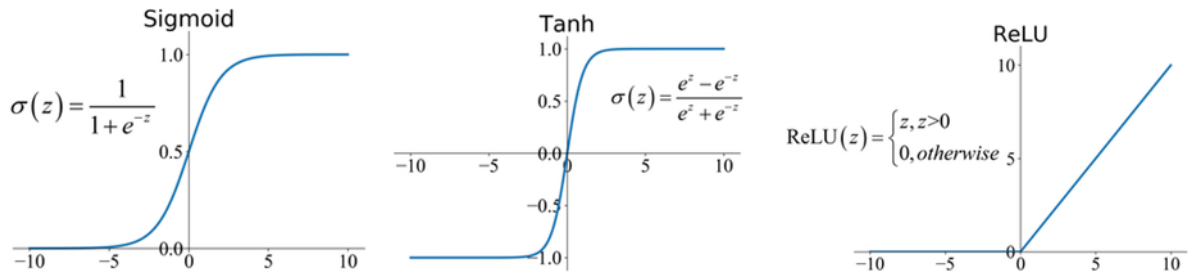
Em seguida, uma função de ativação não linear  $\theta$  é aplicada ao resultado da função linear  $z_j$  para determinar a saída  $a_j$  do neurônio  $j$ , como mostra a Figura 2.5. Desta forma, a soma ponderada  $z_j$  é passada por alguma função de ativação  $\theta$  para produzir a ativação de saída da unidade  $j$ , denotada por  $a_j = \theta(w^T x)$ .



**Figura 2.5** – Representação de um neurônio artificial. Um vetor de entrada  $x$  é linearmente combinado com um vetor de pesos  $w$ , passando por uma função de ativação  $\theta$  para gerar a ativação de saída  $a_j$  de um neurônio  $j$  da camada  $l$ .

A Figura 2.6 mostra algumas das funções de ativação existentes, tais como: função sigmóide ou função de ativação logística,  $\sigma(z) = 1 / (1 + \exp(-z))$ ; função tangente hiperbólica,  $\tanh(z) = 2\sigma(2z) - 1$ ; função ReLU( $z$ ) =  $\max(0, z)$ , dentre outras [LECUN 1989]. Atualmente, a não linearidade ReLU (*Rectified Linear Unit*) ou suas variantes são mais comumente utilizadas. Na Figura 2.5, a saída de um neurônio que utiliza ReLU é calculada como  $a_j = \max(0, z_j)$ , onde  $z_j$  é a somatória na entrada do neurônio. A função de ativação não saturante ReLU tem algumas vantagens sobre os modelos tradicionais de neurônios saturantes sigmoidais, uma vez que evita o desaparecimento e explosão de gradientes [GLOROT e BENGIO 2010] e, assim, reduz significativamente o tempo de treinamento por otimização de pesos com algoritmos baseados em gradiente [SRIVASTAVA *et al.* 2014].



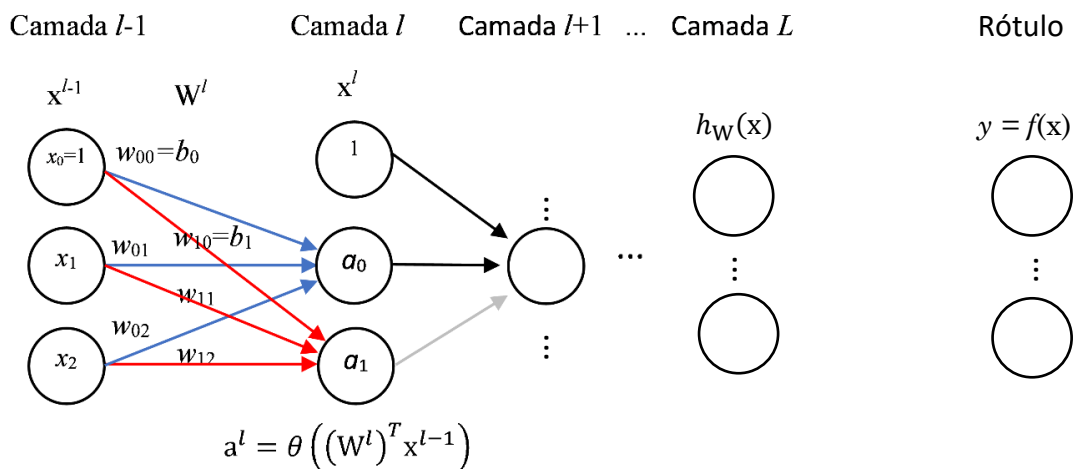


**Figura 2.6** – Funções de ativação saturante sigmóides e não saturante ReLU [NAIR e HINTON 2010].

A saída de uma rede neural multicamada pode ser calculada por propagação direta usando uma abordagem baseada em multiplicações consecutivas de matriz-vetor, como ilustra a **Figura 2.7**. Seja uma camada  $l$  com dimensão  $d^l$  (número de neurônios da camada  $l$ ) com um vetor de entrada  $x^{l-1}$  com dimensão  $d^{l-1} + 1$  (entrada extra igual a 1). Seja  $W^l$  uma matriz de pesos da camada  $l$  com dimensão  $(d^{l-1} + 1) \times d^l$ , ou seja, conexões entre os neurônios da camada anterior  $l - 1$  e todas as unidades na camada  $l$ . Cada coluna  $j$  da matriz contém os pesos associados a um neurônio  $j$  da camada  $l$  (onde a primeira linha contém o viés  $b_j$ ). O peso  $w_{ji}^l$  denota a conexão do neurônio  $i$  na camada  $l - 1$ , com o neurônio  $j$  na camada  $l$ . A saída da camada  $l$  é definida por  $a^l = \theta \left( (W^l)^T x^{l-1} \right)$ . Assim, no exemplo ilustrativo da **Figura 2.7**, a saída da camada  $l$  é dada por:

$$a^l = \theta \left( \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}.$$

A entrada da próxima camada  $l+1$  é dada pelo vetor aumentado com uma entrada de valor 1,  $x^l = \begin{bmatrix} 1 \\ a^l \end{bmatrix}$ . Após a propagação do sinal de entrada por todas as camadas, a saída da rede é denotada pelo vetor  $h_W(x)$ , onde  $h_W$  denota uma função de aproximação com parâmetros  $W$  (dentro de um espaço finito de um conjunto de funções  $H$ ), que se aproxima da função-alvo  $y = f(x)$ .



**Figura 2.7** – Exemplo simplificado de propagação direta para cálculo da saída de uma camada totalmente conectada, usando multiplicações sucessivas de matriz-vetor.

O conjunto total de parâmetros treináveis da rede,  $W = \{W^1, \dots, W^L\}$ , denota todos os pesos  $w$  dos filtros lineares e todos os vieses  $b$  dos neurônios de cada camadas  $l$  da rede, que são aprendidos a partir de pares de amostras rotuladas  $(x_n, y_n)$  do conjunto de dados de treinamento,  $X = \{(x_1, y_1) \dots (x_N, y_N)\}$  [SRIVASTAVA *et al.* 2014]. Os algoritmos de aprendizado ajustam automaticamente os pesos e vieses de uma rede de neurônios artificiais, para que a saída da rede  $h_W(x)$  se aproxime da saída desejada  $y = f(x)$ , considerando todas as amostras rotuladas do conjunto de treinamento,  $(x_i, y_i)$ , onde  $y_i = f(x_i)$ , para  $i = 1, \dots, N$  [ABU-MOSTAFA *et al.* 2012].

O procedimento de aprendizagem envolve a apresentação de um conjunto de pares de padrões de entrada  $x_i$  e saída desejada  $y_i = f(x_i)$ . Primeiro, a rede é inicializada aleatoriamente com um conjunto de pesos  $W$  e usa o vetor de entrada para produzir seu próprio vetor de saída  $h_W(x_i)$ , que inicialmente apresenta uma diferença grande em relação ao *groundtruth*  $y_i$ . Dado um vetor de entrada, o vetor de saída é calculado por uma propagação direta (*forward propagation*) que calcula os níveis de atividade de cada camada, usando os níveis de atividade já calculados nas camadas anteriores. A rede, então, compara o vetor de saída com a saída desejada. Se a diferença é pequena (erro próximo de zero), não ocorre o aprendizado. Caso contrário, os pesos são alterados para reduzir a diferença no próximo passo de treinamento [RUMELHART *et al.* 1986a].

O algoritmo de aprendizado permite ajustar os parâmetros treináveis da rede (pesos e vieses) para minimizar uma função de custo (ou função de erro  $E$ ), obtendo uma saída desejada que se aproxima do alvo. Em outras palavras, o objetivo do algoritmo de aprendizado é encontrar um conjunto de pesos  $W$  que garanta que, para cada vetor de entrada, o vetor de saída produzido pela rede seja o mesmo (ou suficientemente próximo) do vetor de saída desejado (ou seja, o algoritmo de aprendizado minimiza a diferença ou função de erro  $E$ ) [RUMELHART *et al.* 1986].

Uma tarefa é especificada fornecendo o vetor desejado  $y$  das unidades de saída (rótulo) para cada vetor  $x$  das unidades de entrada. Como resultado dos ajustes de peso, unidades ocultas internas passam a representar atributos importantes do domínio específico da tarefa e, dessa forma, as redes aprendem representações internas úteis de dados. O procedimento de aprendizado deve decidir como as unidades ocultas devem ser ativadas, ou seja, quais as representações internas apropriadas para alcançar o comportamento (entrada  $\rightarrow$  saída desejada) [RUMELHART *et al.* 1986].

Se houver um conjunto fixo e finito de dados rotulados (pares de amostras entrada  $\rightarrow$  saída desejada), o erro total ou função de custo  $E(W)$  – usando um determinado conjunto fixo de pesos  $W$  – pode ser calculado comparando-se os vetores de saída preditos e os vetores desejados para cada amostra. Por exemplo, uma função de custo de erro médio quadrático (*Mean Squared Error* – MSE) [WANG e BOVIK 2009] é definida pela Equação 2.2,

$$E(W) = \frac{1}{2n} \sum_x \|f(x) - h_W(x)\|^2 \quad (2.2)$$

onde  $W$  denota todos os pesos  $w$  e todos os vieses  $b$  da rede;  $n$  é o número total de amostras de treinamento;  $y = f(x)$  é o vetor de saída desejado (rótulo), quando um padrão  $x$  é apresentado na entrada; e  $h_W(x)$  é o vetor de saída previsto pela rede, quando o padrão  $x$  é a entrada e  $W$  é o conjunto de parâmetros da rede [LECUN 1989].

### 2.1.1 Gradiente descendente com *backpropagation*

Um otimizador determina como a rede será atualizada com base na função de perda  $E$ . Um algoritmo de aprendizado, com otimização baseada em gradiente descendente e *backpropagation* [LECUN *et al.* 2012], ajusta repetidamente os pesos das conexões na rede, de forma a minimizar a função de erro entre o vetor de saída real da rede e o vetor de saída desejado. Se a função de perda  $E$  que mede o desempenho é diferenciável em relação aos parâmetros ajustáveis do sistema,  $W$ , é possível encontrar um mínimo local de  $E$  usando o aprendizado baseado em gradiente [LECUN *et al.* 1998]. Assim, para minimizar uma função de erro, o algoritmo de aprendizado com *backpropagation* repetidamente calcula o gradiente da função de erro em relação aos pesos, ou seja,  $\partial E(W)/\partial W$ , calculando as derivadas parciais, e então move os pesos na direção oposta do gradiente para minimizar o erro [RUMELHART *et al.* 1986]. Desta forma, o algoritmo de gradiente descendente ajusta o vetor de parâmetros (pesos e vieses)  $W$  iterativamente pela [Equação 2.3](#),

$$W_k = W_{k-1} - \epsilon \frac{\partial E(W)}{\partial W} \quad (2.3)$$

onde  $\epsilon$  é uma constante escalar (taxa de aprendizado). Se a taxa de aprendizado  $\epsilon$  é muito pequena, o algoritmo de aprendizado não converge tão rapidamente. Se a taxa é muito grande, o algoritmo de otimização fica errático e não converge [RUMELHART *et al.* 1986].

Como vimos, o método gradiente descendente envolve o cálculo das derivadas parciais da função de perda em relação aos pesos da rede. O cálculo da derivada é feito usando o algoritmo *backpropagation*. A ideia básica da retropropagação é que os gradientes parciais podem ser calculados eficientemente pela propagação da saída em direção à entrada, usando a regra da cadeia [RUMELHART *et al.* 1986a]. Em outras palavras, para cada instância de treinamento, o algoritmo de aprendizado faz uma previsão por propagação direta e mede o valor da função de erro  $E$ . Em seguida, percorre cada camada por propagação reversa (das camadas superiores para as camadas inferiores), aplicando a regra da cadeia para calcular a contribuição que cada peso da conexão teve na função de perda (medida pela derivada parcial do peso na direção da maior taxa de crescimento da função de erro). Finalmente, os pesos da conexão são ajustados ligeiramente na direção oposta do gradiente para reduzir o erro (etapa de descida do gradiente) [GÉRON 2019]. Maiores informações e detalhes sobre o algoritmo gradiente descendente com *backpropagation* para treinamento de redes neurais tradicionais são encontrados em [RUMELHART *et al.* 1986; LECUN *et al.* 2012].

O objetivo do treinamento de uma rede neural é encontrar um conjunto de pesos e vieses que minimizem a função de erro para todo o conjunto de dados de treinamento. Os algoritmos de aprendizado baseados em gradiente atualizam os parâmetros  $W$  de duas formas. O algoritmo de gradiente estocástico, também chamado de treinamento *online*, consiste em atualizar o vetor de parâmetros usando uma versão ruidosa, onde  $W$  é atualizado com base no cálculo da função de erro de uma única amostra de treinamento (ou um pequeno número de amostras). No método clássico de treinamento em lote (*batch gradient*), os gradientes são acumulados ao longo de todo o conjunto de treinamento e os parâmetros são atualizados após o gradiente médio ter sido calculado. Cada etapa de treinamento calcula a função de erro para um lote ou minilote do conjunto de dados de treinamento e atualiza os pesos e vieses dos parâmetros.



O treinamento é executado para todo o conjunto de dados de treinamento por um número de épocas. Uma época descreve o número de vezes que o algoritmo de aprendizado vê todas as amostras do conjunto de dados [LECUN *et al.* 1998].

A escolha do número de camadas e neurônios, função de ativação, número de épocas de treinamento, tamanho do lote, taxa de aprendizado, dentre outros – conhecidos como hiperparâmetros da rede neural (para distingui-los dos parâmetros treináveis  $W$ , aprendidos pelo algoritmo de aprendizado) – deve ser realizada antes do treinamento. Para evitar vazamentos de informações dos dados de teste, os hiperparâmetros do modelo devem ser ajustados com base no conjunto de validação. Caso o resultado de desempenho medido no conjunto de dados de validação durante o treinamento não seja satisfatório, a rede pode ser treinada novamente usando um novo conjunto de hiperparâmetros. No entanto, deve-se evitar ajustar a configuração do modelo várias vezes com base no desempenho no conjunto de validação, o que pode resultar rapidamente em superajuste ao conjunto de validação. Somente após o treinamento final da rede com os hiperparâmetros fixos, uma avaliação final de desempenho pode ser realizada usando o conjunto de dados de teste para medir quão bem a rede neural generaliza em dados nunca vistos [NIELSEN 2015].

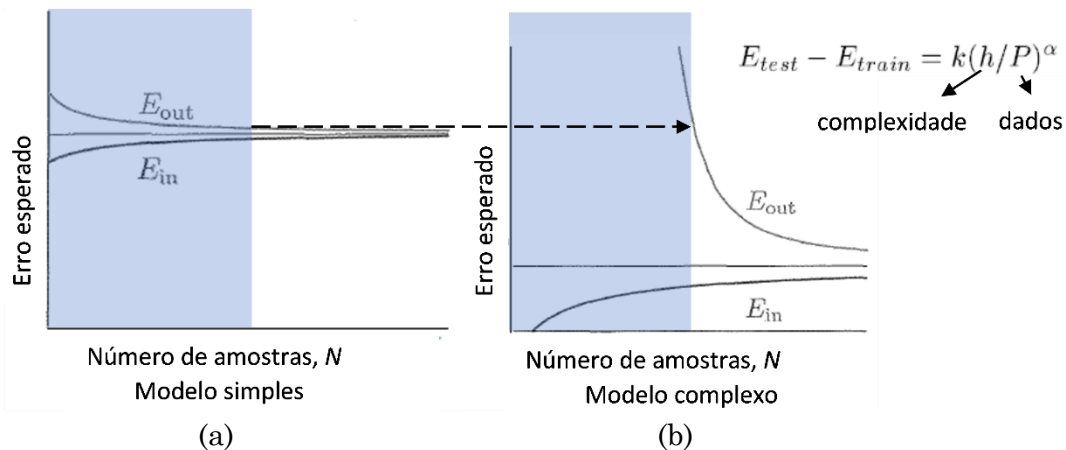
Após a conclusão do treinamento da rede, que define um conjunto fixo de parâmetros  $W$  que minimiza a função de erro  $E_{train}$  no conjunto de treinamento, usamos um conjunto de dados de teste para medir o desempenho real da rede em imagens nunca vistas ( $E_{test}$ ). Se o modelo da rede tem bom desempenho no conjunto de dados de treinamento, mas não generaliza bem para dados de teste, dizemos que ocorreu uma memorização de dados ou um sobreajuste da rede (*overfitting*) aos dados de treinamento.

### 2.1.2 Generalização e sobreajuste

A questão fundamental no aprendizado de máquina é obter um equilíbrio entre otimização (aproximação) e generalização. A otimização refere-se ao processo de aprendizado para ajuste dos parâmetros de um modelo para obter um bom desempenho nos dados de treinamento (aproximação da função  $f(x)$ ), enquanto a generalização se refere a capacidade do modelo treinado de obter um bom desempenho em dados nunca vistos antes. Na prática, o desempenho do modelo de rede em um conjunto de dados de treinamento é de pouco interesse. O objetivo é conseguir uma boa generalização após o treinamento do modelo, medindo o desempenho em um conjunto de amostras disjunto do conjunto de treinamento, chamado de conjunto de teste [CHOLLET 2016].

Quando o modelo obtém bom desempenho no conjunto de treinamento, mas tem um desempenho ruim no conjunto de teste, dizemos que ocorreu um *overfitting* (sobreajuste de dados), ou seja, o modelo aprendeu padrões específicos dos dados de treinamento que são irrelevantes para novos dados. Para evitar que um modelo se ajuste aos dados de treinamento, uma solução é obter um conjunto de dados de treinamento maior. Quando isso não é possível, um aumento artificial de dados (*Data Augmentation* – DA) permite gerar mais dados de treinamento por meio de transformações aleatórias (como translação, rotação, reflexão, escala, etc.) a partir de amostras de treinamento existentes [RUSSAKOVSKY *et al.* 2015].

Um modelo treinado com mais dados irá generalizar melhor [CHOLLET 2016]. A Figura 2.8 mostra que o *gap* entre a taxa de erro no conjunto de teste,  $E_{test}$ , e a taxa de erro no conjunto de treinamento,  $E_{train}$ , diminui quando o número de amostras de treinamento aumenta<sup>3</sup>. O *gap* é dado aproximadamente por  $E_{test} - E_{train} = k(h/P)^\alpha$ , onde  $P$  é o número de amostras de treinamento,  $h$  é a medida de capacidade efetiva ou complexidade da máquina,  $\alpha$  é um número entre 0,5 e 1, e  $k$  é uma constante. Além disso, quando a capacidade  $h$  aumenta,  $E_{train}$  diminui. Portanto, quando a capacidade  $h$  aumenta há um *tradeoff* entre diminuição de  $E_{train}$  e aumento do *gap*, com um valor ótimo de capacidade  $h$  que adquire o menor erro de generalização  $E_{test}$  [LECUN *et al.* 1998].



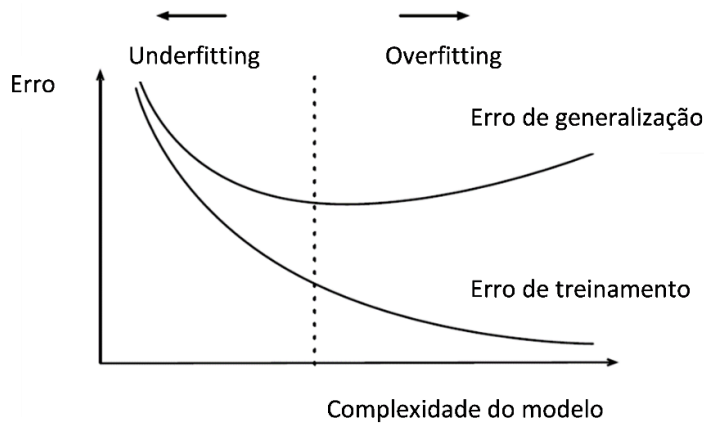
**Figura 2.8** – *Gap* entre erro de treinamento e teste de acordo com a complexidade do modelo de rede e quantidade de dados de treinamento. Adaptado de [ABU-MOSTAFA *et al.* 2012].

Os erros esperados são funções do número de amostras  $N$  e são chamados de curvas de aprendizado do modelo. Na Figura 2.8, as curvas de aprendizado para um modelo de aprendizado simples e um modelo complexo resumem o comportamento dos erros  $E_{train} = E_{in}$  e  $E_{out} = E_{test}$ , conforme varia o tamanho do conjunto de dados de treinamento. As curvas de aprendizado do modelo simples convergem mais rapidamente, mas têm um desempenho pior do que o modelo complexo. Para modelos simples e complexos, a curva de aprendizado  $E_{out}$  diminui, enquanto a curva  $E_{in}$  aumenta com o crescimento de  $N$ . Na Figura 2.8b, o sobreajuste ocorre para um intervalo de poucas amostras  $N$  na região sombreada, porque escolhendo um modelo mais complexo que tem  $E_{in}$  menor, temos  $E_{out}$  maior que o modelo mais simples [ABU-MOSTAFA *et al.* 2012]. Neste caso, o modelo mais complexo exige uma quantidade maior de dados para evitar *overfitting*.

Outra forma de evitar o sobreajuste é reduzir o tamanho do modelo, ou seja, o número de parâmetros treináveis no modelo (que é determinado pelo número de camadas e pelo número de unidades por camada). No aprendizado profundo, o número de parâmetros que podem ser aprendidos geralmente é chamado de capacidade do modelo. Intuitivamente, um modelo com mais parâmetros têm mais capacidade de memorização, sem capacidade de generalização [CHOLLET 2016]. Quando o número de parâmetros livres é grande, o classificador tende a se adaptar a detalhes específicos da base de treinamento, o que pode causar uma redução de desempenho no conjunto de teste. Na Figura 2.9, à medida que a complexidade do modelo da rede aumenta,  $E_{train}$  diminui, mas  $E_{test}$  aumenta. Portanto, quando a complexidade aumenta, há um custo-benefício entre a diminuição de  $E_{train}$  e o aumento do *gap*, com um valor ótimo de complexidade que adquire o erro de generalização  $E_{test}$  mais baixo, como mostra a Figura 2.9 [LECUN *et al.* 1998].

<sup>3</sup> A taxa de erro  $E_{test}$  também é chamada de erro de generalização ou erro fora da amostra de treinamento,  $E_{out}$  (*out-of-sample error*). A taxa de erro  $E_{train}$  é chamada de erro de treinamento ou erro dentro da amostra de treinamento  $E_{in}$  (*in-sample error*) [GÉRON 2019].

O subajuste de dados de treinamento (*underfitting*) é o oposto do *overfitting*, quando o modelo é muito simples para aprender a estrutura dos dados de treinamento. Consequentemente é necessário encontrar um equilíbrio entre encontrar um modelo com complexidade suficiente para ajustar os dados de treinamento e manter o modelo simples o suficiente para garantir uma boa generalização [GÉRON 2019].



**Figura 2.9** – Influência da complexidade do modelo no ajuste aos dados de treinamento e capacidade de generalização (*underfitting* e *overfitting*). Adaptado de [ZHANG *et al.* 2020].

Como vimos até agora, o desempenho da generalização aumenta à medida que a quantidade de dados aumenta e o número de parâmetros livres na rede diminui [LECUN 989]. Além de aumentar a quantidade de dados e diminuir a complexidade do modelo, existem outras técnicas de regularização para diminuir o sobreajuste de dados. *Dropout* [HINTON *et al.* 2012] é uma das técnicas de regularização mais eficazes para redes neurais profundas, que consiste em descartar aleatoriamente as unidades de uma camada durante o treinamento [CHOLLET 2016].

Quando uma rede neural profunda é treinada em um pequeno conjunto de dados de treinamento em redes complexas, ela normalmente tem um desempenho ruim em dados de teste, porque os detectores de atributos (unidade ou neurônios) foram ajustados para funcionar bem nos dados de treinamento. Este sobreajuste é bastante reduzido omitindo-se aleatoriamente metade dos neurônios em cada etapa de treinamento para evitar coadaptações complexas nos dados de treinamento [SRIVASTAVA *et al.* 2014]. Em cada etapa de treinamento, cada unidade de uma camada oculta é omitida aleatoriamente da rede com uma probabilidade de 0.5, de modo que uma unidade oculta não pode contar com a presença de outras unidades ocultas, reduzindo a complexidade da rede.

Outra forma de reduzir o erro no conjunto de dados de teste é calcular a média das previsões geradas por redes diferentes, treinadas separadamente. Consequentemente, uma forma de entender o *dropout* é perceber que uma rede neural única é gerada a cada etapa do treinamento, com um total de  $2^N$  redes possíveis (onde  $N$  é o número total de neurônios que podem ser eliminados). A rede neural resultante pode ser interpretada como um conjunto médio de todas essas redes neurais menores [GÉRON 2019]. Desta forma, o *dropout* aleatório pode ser visto como um treinamento de um grande número de redes diferentes em tempo razoável. No momento do teste, a rede média que contém todas as unidades ocultas é usada, mas seus pesos de saída são reduzidos pela metade (por um fator igual a taxa de *dropout*) para compensar o fato de que o dobro de unidades estão ativas na fase de teste em relação à fase de treinamento [SRIVASTAVA *et al.* 2014].

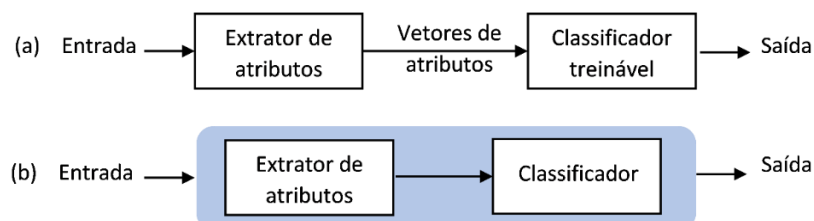
Um grande número de parâmetros das redes neurais multicamadas totalmente conectadas e não estruturadas aumenta a complexidade do modelo e, portanto, requer um conjunto de treinamento maior. Em um sistema típico de aprendizado profundo com redes totalmente conectadas, pode haver centenas de milhões de pesos ajustáveis, sendo necessário centenas de milhões de dados rotulados para treinar a máquina. Desta forma, quanto mais profunda a rede, mais dados são necessários e maior o tempo de treinamento. Porém, a principal deficiência de redes não estruturadas aplicadas a imagens é que elas não têm invariância com relação a translações ou distorções locais das entradas. Assim, seria necessário um grande número de amostras de treinamento para cobrir o espaço de possíveis variações.

Além disso, nas arquiteturas totalmente conectadas, a topologia da entrada é totalmente ignorada, desconsiderando o fato de que as imagens têm uma estrutura local 2D, com *pixels* espacialmente próximos altamente correlacionados. Por estes motivos, foram desenvolvidas as redes neurais convolucionais (CNN), que têm algum grau de invariância ao deslocamento, escala e distorção das entradas. As CNNs são um tipo específico de rede neural profunda com neurônios dispostos em uma grade 2D, com uma quantidade menor de parâmetros treináveis do que as redes totalmente conectadas e, conseqüentemente, mais fácil de treinar com menos sobrejeste de dados [LECUN *et al.* 1998].

## 2.2 Redes neurais convolucionais

Os métodos tradicionais de aprendizado de máquina para reconhecimento de padrões dividem o sistema em dois módulos principais, como mostrado na [Figura 2.10a](#). O extrator de atributos, projetado especificamente para uma tarefa, transforma os padrões de entrada para que possam ser representados por vetores de atributos (características) de baixa dimensionalidade. Em seguida, um classificador treinável categoriza os vetores de atributos resultantes em classes. Geralmente, técnicas tradicionais de aprendizado de máquina são usadas como classificador, tais como KNN, SVM, dentre outros. No entanto, redes neurais multicamadas totalmente conectadas (RNA) também podem ser usadas.

Devido ao progresso em aprendizado de máquina, a extração manual de atributos (*feature engineering*) foi substituída por redes neurais convolucionais (CNNs), com aprendizado automático de ponta a ponta, que operam diretamente nos *pixels* das imagens de entrada, para melhorar os sistemas de classificação de imagens ([Figura 2.10b](#)). Neste esquema, o extrator de atributos é aprendido junto com uma rede neural classificadora, permitindo tratar entradas de alta dimensionalidade e gerar funções de decisão complexas quando alimentadas com grandes conjuntos de dados [LECUN *et al.* 1998].



**Figura 2.10** – *Aprendizado de máquina tradicional: extrator de atributos e classificador tradicional treinável (a); Rede neural convolucional de aprendizado profundo treinada de ponta a ponta (b). Adaptado de [LECUN *et al.* 1998].*

Desde o início dos anos 2000, redes neurais convolucionais (CNN), também chamadas de *convnets*, têm sido aplicadas com sucesso na classificação, detecção e segmentação semântica de imagens. Redes neurais convolucionais profundas (DCNNs) usam uma arquitetura bem-adaptada para classificar imagens, uma vez que levam em conta a estrutura espacial da imagem. Em outras palavras, as CNNs são projetadas para processar dados que vêm na forma de múltiplos *arrays*, como por exemplo, uma imagem colorida composta de três *arrays* 2D contendo intensidades de *pixels* dos três canais de cores RGB [RAWAT e WANG 2017]. Desta forma, a rede CNN é alimentada diretamente com informações de *pixels* das imagens (informação de baixo nível), em vez de vetores de atributos. Uma única CNN aprende toda a operação de reconhecimento, desde a entrada da imagem normalizada até a classificação final [LECUN *et al.* 1990].

Três tipos principais de camadas são usados para construir a arquitetura de uma CNN: camada convolucional, camada de agrupamento com subamostragem (*pooling*) e camada totalmente conectada. As CNNs consistem em uma sequência alternada de várias camadas convolucionais e de subamostragem, seguida por camadas totalmente conectadas. As camadas convolucionais atuam como extratores de atributos das imagens de entrada, cuja dimensionalidade é sucessivamente reduzida pelas camadas de *pooling*, enquanto as camadas totalmente conectadas atuam como classificadores. As camadas totalmente conectadas exploram os atributos de alto nível aprendidos pelas camadas anteriores, com objetivo de classificar as imagens de entrada em classes predefinidas.

## 2.2.1 Arquitetura de rede neural convolucional

A arquitetura LeNet [LECUN 1989; LECUN *et al.* 1989; LECUN *et al.* 1990], proposta por Yann LeCun em 1988, foi uma das primeiras arquiteturas de CNN utilizada para reconhecimento de caracteres, tais como código postal e dígitos numéricos. Novas arquiteturas foram propostas posteriormente, embora as versões de CNN melhoradas compartilhem os conceitos fundamentais da LeNet-5. A arquitetura LeNet-5 [LECUN *et al.* 1998] mostrada na Figura 2.11 compreende 7 camadas, sem contar a entrada, todas contendo parâmetros treináveis (pesos e vieses). As camadas convolucionais são nomeadas Cx, as camadas de agrupamento com subamostragem (*pooling*) são nomeadas Sx e as camadas totalmente conectadas são nomeadas Fx, onde x é o índice da camada.

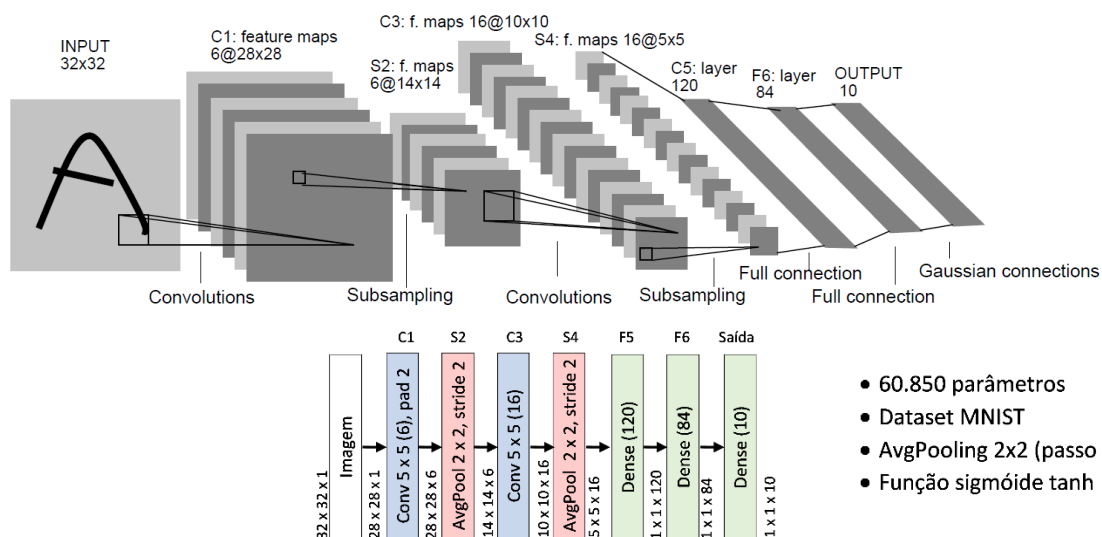


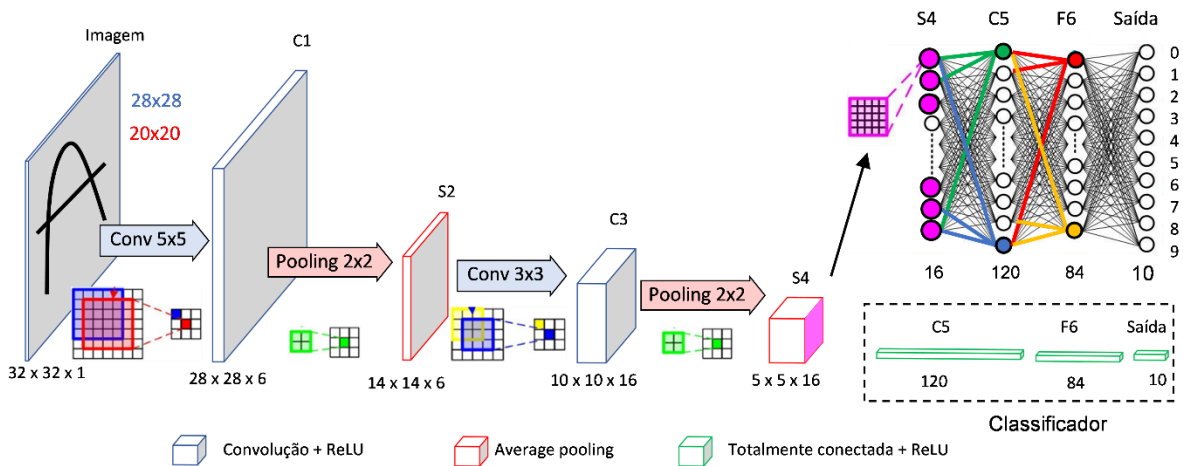
Figura 2.11 – Arquitetura LeNet-5 com 7 camadas [LECUN *et al.* 1998; ZHANG *et al.* 2020].



As redes convolucionais combinam três ideias para assegurar algum grau de invariância ao deslocamento, escala e distorção: campos receptivos locais, pesos compartilhados e subamostragem espacial, como descrito nas próximas seções.

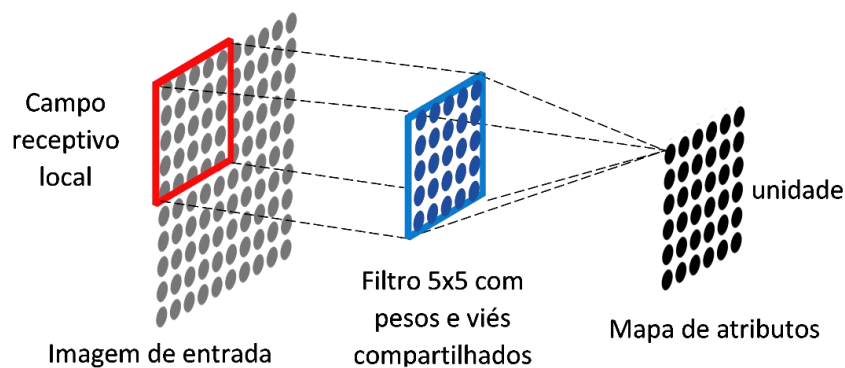
### 2.2.1.1 Camada convolucional

A entrada da rede LeNet-5 na [Figura 2.12](#) é uma imagem monocromática de  $32 \times 32$  *pixels* de caracteres de no máximo  $20 \times 20$  *pixels* centralizados em um campo de  $28 \times 28$ . Desta forma, a CNN tem  $32 \times 32$  neurônios ou unidades de entrada, cujos valores correspondem às intensidades normalizadas de  $32 \times 32$  *pixels* da imagem do conjunto de dados MNIST [[LECUN et al. 1998](#)].



**Figura 2.12** – Mapas de atributos da rede LeNet-5.

A primeira camada convolucional C1 tem 6 mapas de atributos (*feature maps*) com valores da ativação de  $28 \times 28$  neurônios ou unidades ocultas. Os *pixels* de entrada estão conectados aos neurônios ocultos de cada mapa de atributos da primeira camada C1, porém cada *pixel* de entrada não será conectado a todos os neurônios ocultos. Em vez disso, cada unidade de cada mapa de atributos é conectada a uma vizinhança de  $5 \times 5$  *pixels* na entrada. Essa região na imagem de entrada é chamada de campo receptivo local para aquele neurônio oculto, como mostra o [Figura 2.13](#). Em seguida, o campo receptivo local é deslizado por toda a imagem de entrada, para corresponder a um neurônio oculto diferente na primeira camada. O campo receptivo local é deslizado com um passo de um ou mais *pixels* à direita (e para baixo), para conectar a um segundo neurônio oculto [[LECUN 1989](#)].



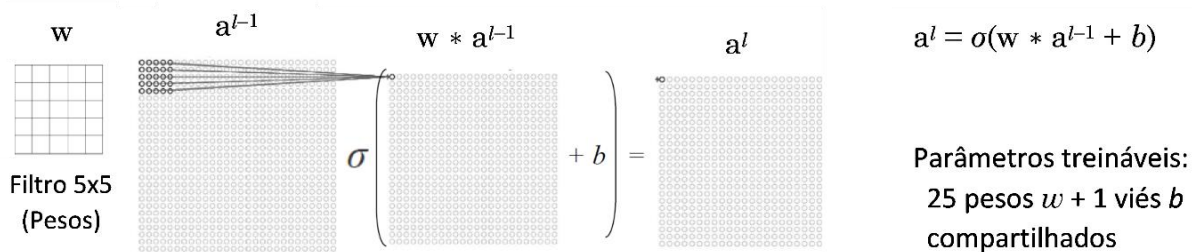
**Figura 2.13** – Campo receptivo local com filtro de pesos compartilhados  $5 \times 5$ , conectado ao primeiro neurônio oculto de um dos mapas de atributos da primeira camada C1.

Cada neurônio oculto ou unidade de um mapa de atributos da camada C1 tem um viés e uma matriz de pesos  $5 \times 5$  conectada ao seu campo receptivo local. Com os campos receptivos locais, os neurônios da primeira camada podem extrair atributos visuais elementares que aparecem em toda a imagem, como por exemplo, bordas orientadas, pontos finais e cantos. Esses atributos serão combinados pelas camadas subsequentes para detectar atributos de ordem superior. Os detectores de atributos locais úteis em uma parte da imagem provavelmente serão úteis em toda a imagem. Portanto, todos os neurônios de um mapa de atributos detectam um único tipo de atributo local em diferentes partes da imagem de entrada. Sendo assim, os mesmos pesos e um viés compartilhados, que definem um *kernel* ou filtro, são usados para calcular os  $28 \times 28$  neurônios ocultos que definem um dos mapas de atributos da camada C1. O compartilhamento de pesos permite extrair atributos invariantes ao deslocamento, ou seja, deslocar a imagem de entrada deslocará o resultado no mapa de atributos de saída, mas a forma será inalterada [LECUN *et al.* 1990].

Na Figura 2.14, um mapa de atributos da camada  $l$  é gerado pela convolução com um pequeno *kernel*  $w$ , seguido por um viés aditivo  $b$  e uma função de ativação  $\sigma$ , dado pela Equação 2.4,

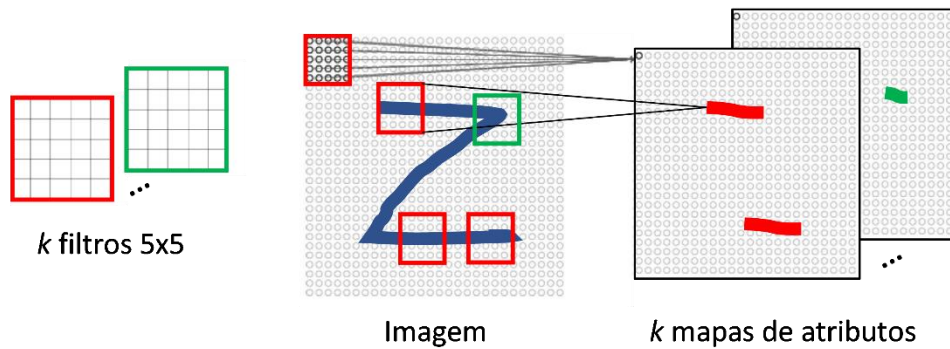
$$a^l = \sigma(w * a^{l-1} + b), \quad (2.4)$$

onde  $a^l$  denota o conjunto de ativações de saída de um mapa de atributos da camada  $l$ ,  $a^{l-1}$  é o conjunto de ativações de entrada,  $w$  é a matriz de pesos compartilhados pelo mapa de atributos,  $b$  é o viés compartilhado, e  $*$  é chamado de operação de convolução. Os valores resultantes após a operação de convolução passam por uma função de ativação  $\sigma$ , como a função sigmóide *tanh*, sendo atualmente mais usada a função ReLU (*Rectified Linear Units*) [NAIR e HINTON 2010]. Desta forma, a função executada por um mapa de atributos pode ser interpretada como uma convolução não linear com um filtro ou *kernel*  $5 \times 5$  [LECUN *et al.* 1989]. Por este motivo, a rede é chamada rede neural convolucional.



**Figura 2.14** – Operação de convolução não linear da entrada com um filtro  $5 \times 5$ .

Uma camada convolucional completa é composta por vários mapas de atributos com diferentes conjuntos de detectores de atributos (*kernels* com pesos e viés compartilhados para cada mapa de atributos), de modo que vários atributos locais possam ser extraídos em qualquer localização da imagem (ou mapa de atributos) de entrada. O viés e pesos compartilhados de cada mapa de atributos são aprendidos pela rede, como se um neurônio oculto particular estivesse aprendendo a analisar seu campo receptivo local específico, como mostra a Figura 2.15 [LECUN 1989].

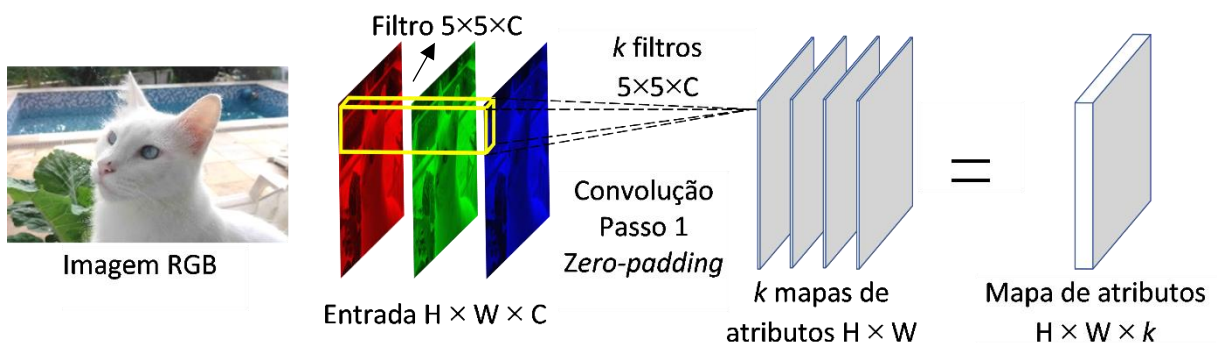


**Figura 2.15** – Camada de convolução com  $k$  mapas de atributos para detecção de vários tipos de atributos. Cada unidade de um mapa de atributos compartilha os mesmos pesos do filtro correspondente, detectando o mesmo atributo em locais diferentes.

A primeira camada convolucional C1 tem seis mapas de atributos. Desta forma, a rede pode detectar seis tipos diferentes de atributos em toda a imagem. Cada mapa de atributos é definido por um conjunto de  $5 \times 5 = 25$  pesos compartilhados, além de um único viés compartilhado, ou seja, requer 26 parâmetros treináveis. No total, a camada C1 tem 156 parâmetros treináveis (26 x 6 mapas) e 122.304 conexões. Uma grande vantagem do compartilhamento de pesos e viés é a redução do número de parâmetros envolvidos em uma rede convolucional, que resulta em um treinamento mais rápido e permite construir redes convolucionais mais profundas [LECUN 1989].

Na rede LeNet-5, imagens monocromáticas 2D são admitidas como entrada de uma camada convolucional. Para cada filtro de convolução 2D, um mapa de atributos é gerado na saída da camada. Quando as camadas convolucionais recebem como entrada um conjunto 3D, também chamado de volume, cada filtro se estende por toda a profundidade do volume de entrada. Por exemplo, para imagens RGB com  $C = 3$  canais de profundidade, o filtro 3D da primeira camada convolucional tem tamanho  $5 \times 5 \times C$ , como mostra a Figura 2.16.

Uma camada convolucional de entrada ou intermediária contém um conjunto de  $k$  filtros (2D ou 3D, dependendo da dimensionalidade da entrada) para detectar  $k$  tipos de atributos na entrada. Independentemente da dimensionalidade do filtro, cada filtro gera um mapa de atributos 2D. Assim, a saída de uma camada convolucional gera um conjunto de  $k$  mapas de atributos 2D, ou seja, um mapa de atributos 3D como entrada da próxima camada [LECUN 1989].

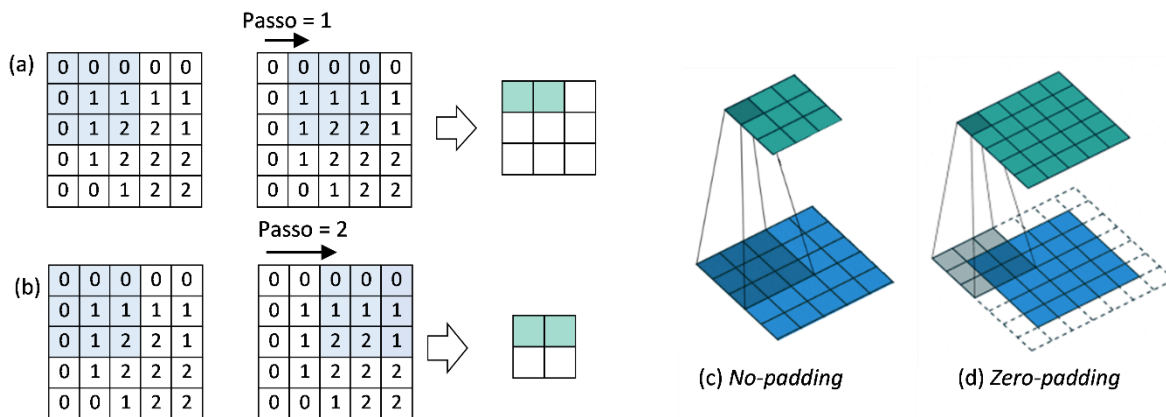


**Figura 2.16** – Convolução de  $k$  filtros 3D com a imagem de entrada RGB, gerando um conjunto de  $k$  mapas de atributos 2D na saída de uma camada de convolução.



Existem três parâmetros que controlam o tamanho do volume na saída da camada convolucional: número de filtros (*depth*), passo (*stride*) e preenchimento (*padding*). A profundidade do volume resultante é igual ao número de filtros utilizados. Cada um desses filtros será responsável por extrair atributos diferentes no volume de entrada. Portanto, quanto maior o número de filtros, maior o número de mapas de atributos extraídos, porém a complexidade computacional, tempo de execução e uso de memória também serão maiores [LECUN 1989].

A altura e largura da saída de cada camada dependem do passo e preenchimento. O *passo* (*stride*) especifica o tamanho do salto na operação de convolução, como ilustrado na Figura 2.17a-b. Quanto maior o valor do passo, menor será a altura e largura da saída, porém informações importantes podem ser perdidas. A operação de preenchimento (*zero-padding*) consiste em preencher com zeros a borda do volume de entrada. Assumindo passo  $s = 1$ , tamanho da entrada  $m \times n$ , *kernel* de convolução  $k \times k$ , e adicionando  $p$  linhas de preenchimento na parte superior e inferior, e  $p$  colunas à esquerda e à direita, o tamanho da saída pode ser calculado por  $\lfloor (m - k + 2p + s) / s \rfloor \times \lfloor (n - k + 2p + s) / s \rfloor$ . Desta forma, essas operações permitem controlar a altura e largura da saída para que fiquem iguais à entrada, como mostra a Figura 2.17d.

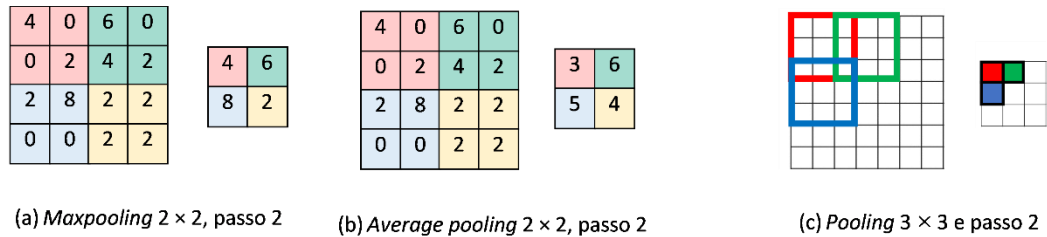


**Figura 2.17** – Convolução com passo unitário (a); convolução com passo 2 (b); sem zero-padding reduz tamanho na saída (c); zero-padding igual a 1 mantém o tamanho da entrada após convolução com filtro  $3 \times 3$  e passo 1 (d) [DUMOULIN e VISIN 2016].

### 2.2.1.2 Camada de subamostragem

Redes neurais convolucionais contêm camadas de agrupamento com subamostragem (*pooling*) usadas após uma camada convolucional para simplificar e condensar os atributos de saída. Na operação de *pooling*, os valores de uma determinada região do mapa de atributos de entrada (campos receptivos) são substituídos pelo valor máximo (*maxpooling*) ou média (*average-pooling*), como ilustra a Figura 2.18a e b.

Para obter uma subamostragem do mapa de atributos de entrada, geralmente os campos receptivos  $2 \times 2$  não são sobrepostos usando-se passo 2 e, conseqüentemente, os mapas de atributos de saída têm metade do número de linhas e colunas dos mapas de atributos da camada anterior. Algumas redes CNN também obtêm uma subamostragem com campos receptivos sobrepostos, por exemplo, usando campo receptivo  $3 \times 3$  e passo 2 (Figura 2.18c) ou realizam a operação de *pooling* sem subamostragem com passo 1.



**Figura 2.18** – Maxpooling (a) e average-pooling (b) com campo receptivo não sobreposto  $2 \times 2$  (passo 2); subamostragem sobreposta com campo  $3 \times 3$  e passo 2 (c).

Na rede LeNet-5 (Figura 2.12), a camada de subamostragem S2 tem 6 mapas de atributos de  $28 \times 28$  neurônios na entrada. Como o *average-pooling* é aplicado para cada mapa de atributos separadamente, a profundidade da entrada não é alterada. Portanto, a camada S2 gera 6 mapas de atributos de tamanho  $14 \times 14$ , onde cada unidade de cada mapa de atributos de S2 é conectada a uma vizinhança correspondente  $2 \times 2$  no mapa C1.

Especificamente na LeNet-5, cada unidade em S2 calcula a média de suas quatro entradas, multiplica por um coeficiente treinável e acrescenta um viés treinável. O resultado é passado por uma função de ativação sigmoideal. Assim, a camada S2 possui 12 parâmetros treináveis (um peso e um viés para cada mapa de atributo) e 5.880 conexões [LECUN *et al.* 1998]. Nas CNNs mais recentes, não é utilizado nenhum parâmetro treinável, nem função de ativação na operação de *pooling*.

A operação de *pooling* encontra um determinado atributo em qualquer lugar da imagem, porém, sem a informação de posição exata do atributo. Como a localização precisa de um atributo não é tão importante para a classificação quanto sua localização aproximada em relação a outros atributos, não há problema perder algumas informações de posição. No entanto, a posição aproximada deve ser preservada para permitir que as próximas camadas detectem atributos de ordem superior [LECUN 1989]. A vantagem da subamostragem para redução da resolução espacial do mapa de atributos é a diminuição da sensibilidade da saída a deslocamentos e distorções, e, principalmente, redução do número de parâmetros necessários nas camadas posteriores.

Nas arquiteturas CNN, camadas de convoluções e *pooling* são sucessivamente alternadas, aumentando o número de mapas de atributos à medida que a resolução espacial é diminuída. Na LeNet-5, a camada convolucional C3, com *kernel*  $3 \times 3$ , conectada à camada S2, produz uma saída de 16 mapas de atributos  $10 \times 10$ , com 1.516 parâmetros treináveis e 151.600 conexões. A camada S4 (*average-pooling*) reduz para 16 mapas de atributos de tamanho  $5 \times 5$ . Cada unidade em cada mapa de atributos é conectada a uma vizinhança  $2 \times 2$  no mapa de atributos correspondente em C3. A camada S4 possui 32 parâmetros treináveis (um peso e um viés para cada mapa de atributos) e 2.000 conexões.

A última camada convolucional C5 usa 120 filtros 3D de tamanho  $5 \times 5$ , que também aplaina o volume de ativação de entrada em um vetor, gerando 120 “mapas” de atributos de uma unidade. Cada unidade é conectada a uma vizinhança de  $5 \times 5$  em todos os 16 mapas de atributos do S4. Como o tamanho de S4 também é  $5 \times 5$ , o tamanho dos mapas de atributos de C5 é  $1 \times 1$ : isso equivale a uma conexão completa entre S4 e C5, como mostra a Figura 2.12. A camada C5 tem 48.120 conexões treináveis (25 x 16 pesos e um viés para cada mapa de atributos) [LECUN 1989].

### 2.2.1.3 Camadas totalmente conectadas

Na arquitetura LeNet-5 mostrada na [Figura 2.12](#), a saída da camada C5, após uma sequência alternada de camadas convolucionais e de subamostragem, representa os atributos de alto nível extraídos da imagem de entrada. As camadas classificadoras finais F6 e F7 (camada de saída) são totalmente conectadas, o que significa que todos os neurônios da camada anterior estão conectados a todos os neurônios da camada seguinte. As camadas totalmente conectadas são exatamente iguais a uma rede neural artificial convencional, que usa a função de ativação *softmax* ([Apêndice D.1](#)) na última camada de saída. O objetivo das camadas totalmente conectadas é utilizar os atributos de alto nível da camada C5 para classificar a imagem em uma classe predeterminada.

A camada F6 contém 84 unidades ou neurônios totalmente conectados ao C5 e tem 10.164 parâmetros treináveis (84 x 120 pesos e 84 vieses). Finalmente, a camada F7 de saída consiste em 10 unidades, que correspondem ao número de classes, ou seja, aos 10 valores possíveis para dígitos manuscritos do conjunto de dados MNIST ('0', '1', '2', ..., '9') [[LECUN et al. 1998](#)]. Os 84 neurônios da camada F6 são totalmente conectados aos 10 neurônios da camada F7. Por fim, a rede LeNet-5 contém no total 340.908 conexões, mas apenas 60.850 parâmetros livres treináveis por causa do compartilhamento de pesos.

Uma arquitetura de rede neural artificial RNA, totalmente conectada, com poder discriminativo suficiente para determinadas tarefas de reconhecimento de objetos, tem muitos parâmetros para ser capaz de generalizar corretamente [[LECUN et al. 1990](#)]. Nas redes neurais convolucionais, o compartilhamento de pesos dos filtros, além de detectar um mesmo atributo em qualquer posição da imagem (com invariância ao deslocamento), diminui o número de parâmetros livres. Desta forma, as CNNs reduzem a complexidade do modelo e a quantidade de dados de treinamento necessária para obter um certo nível de desempenho de generalização; e, portanto, reduzem o *gap* entre o erro de treinamento e teste [[LECUN et al. 1998](#)].

Em contraste com a RNA, cujas exigências de tempo de treinamento são impraticáveis em alguns problemas com muitos parâmetros, a CNN pode aprender problemas complexos rapidamente por causa do compartilhamento de pesos, que diminui o número de parâmetros livres e aumenta o desempenho de generalização, desde que existam conjuntos de dados disponíveis suficientes (isto é, centenas até milhares de dados de treinamento, dependendo da complexidade do problema) [[LECUN 1989](#)].

As camadas convolucionais reduzem drasticamente a complexidade do modelo, mas também reduzem a possibilidade de sobreajuste, de modo que o *dropout* é muito menos vantajoso em camadas convolucionais, sendo usado apenas nas camadas finais totalmente conectadas das arquiteturas CNN [[SRIVASTAVA et al. 2014](#)].

A rede neural convolucional também é treinada usando um algoritmo de gradiente descendente estocástico (SGD), usando um algoritmo modificado de propagação reversa (*backpropagation*) para calcular os gradientes da função de custo em relação a todos os pesos em todas as camadas da rede convolucional. O algoritmo padrão foi modificado para levar em consideração o compartilhamento de pesos.

Como todos os pesos dos filtros das redes convolucionais são aprendidos de ponta a ponta, o extrator de atributos é sintetizado pela própria rede neural, cujos atributos são classificados pelas camadas totalmente conectadas.

## 2.3 Arquiteturas de classificação de imagens

Com a evolução das redes neurais, algumas práticas permitiram melhorar o desempenho das CNNs em tarefas de reconhecimento de objetos: (1) aumentar o conjunto de dados de treinamento; (2) desenvolver modelos mais complexos, com maior capacidade de aprendizado; (3) usar técnicas de regularização para evitar sobreajuste de dados.

Até 2012, as redes neurais eram pouco complexas e usavam conjuntos de dados pequenos; por isso, foram superadas por outros métodos de aprendizado de máquina tradicionais, como SVM. Com a disponibilidade de um grande conjunto de dados, como ImageNet [DENG *et al.* 2009], com mais de 15 milhões de imagens de alta resolução rotuladas em mais de 22.000 categorias, vários modelos de CNN com grande capacidade de aprendizado e maior desempenho de generalização foram desenvolvidos, mas todos compartilham as características principais da LeNet-5 [LECUN *et al.* 1998]. Para grandes conjuntos de dados, a tendência foi aumentar a capacidade de redes neurais convolucionais, controlando a profundidade (número de camadas) e a largura da rede (quantidade de filtros em cada camada). Métodos de regularização, como *dropout*, foram utilizados para resolver o problema de sobreajuste de dados [SRIVASTAVA *et al.* 2014].

Nesta seção, descrevemos algumas CNNs para classificação de imagens, tais como, AlexNet (2012) [KRIZHEVSKY *et al.* 2012], VGG (2014) [SIMONYAN e ZISSERMAN 2015], GoogleNet (2014) [SZEGEDY *et al.* 2015] e outras versões de redes Inception, redes residuais ResNet (2015) [HE *et al.* 2016a], WideResNet (2016) [ZAGORUYKO e KOMODAKIS 2016], Xception (2017) [CHOLLET 2017] e DenseNet (2018) [HUANG *et al.* 2017]. Quase todas estas arquiteturas são disponibilizadas com pesos pré-treinados no ImageNet, fornecendo reconhecimento preciso para este domínio de problema específico.

### 2.3.1 Modelo AlexNet

Em 2012, foi criada a rede AlexNet [KRIZHEVSKY *et al.* 2012], mais profunda que LeNet-5, com 60 milhões de parâmetros e 650.000 neurônios, como mostra a Figura 2.19. A rede LeNet-5 [LECUN *et al.* 1998] na Figura 2.11 tem duas camadas de convolução para extração de atributos, alternadas com operações de subamostragem espacial. Essas ideias foram refinadas na arquitetura AlexNet, onde as camadas de convolução são repetidas várias vezes entre as camadas de *maxpooling*, permitindo que a rede aprenda atributos mais ricos. A rede AlexNet consiste em oito camadas: cinco camadas convolucionais, com três seguidas por camadas de *maxpooling*, além de três camadas totalmente conectadas, sendo uma camada de saída *softmax* de 1.000 vias. Esta rede foi treinada para classificar um subconjunto de ImageNet, contendo 1,2 milhões de imagens de alta resolução rotuladas em 1.000 classes, na competição ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*) [BERG *et al.* 2010].

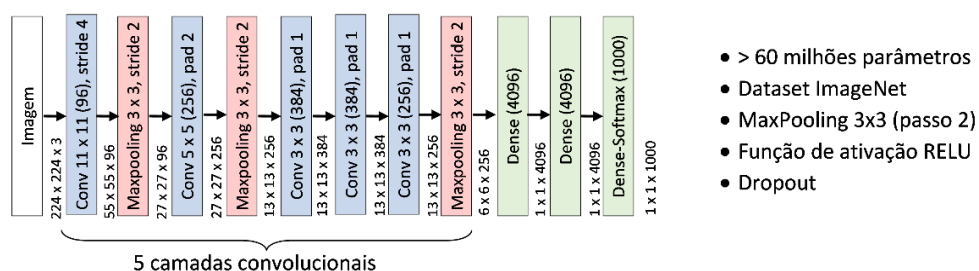
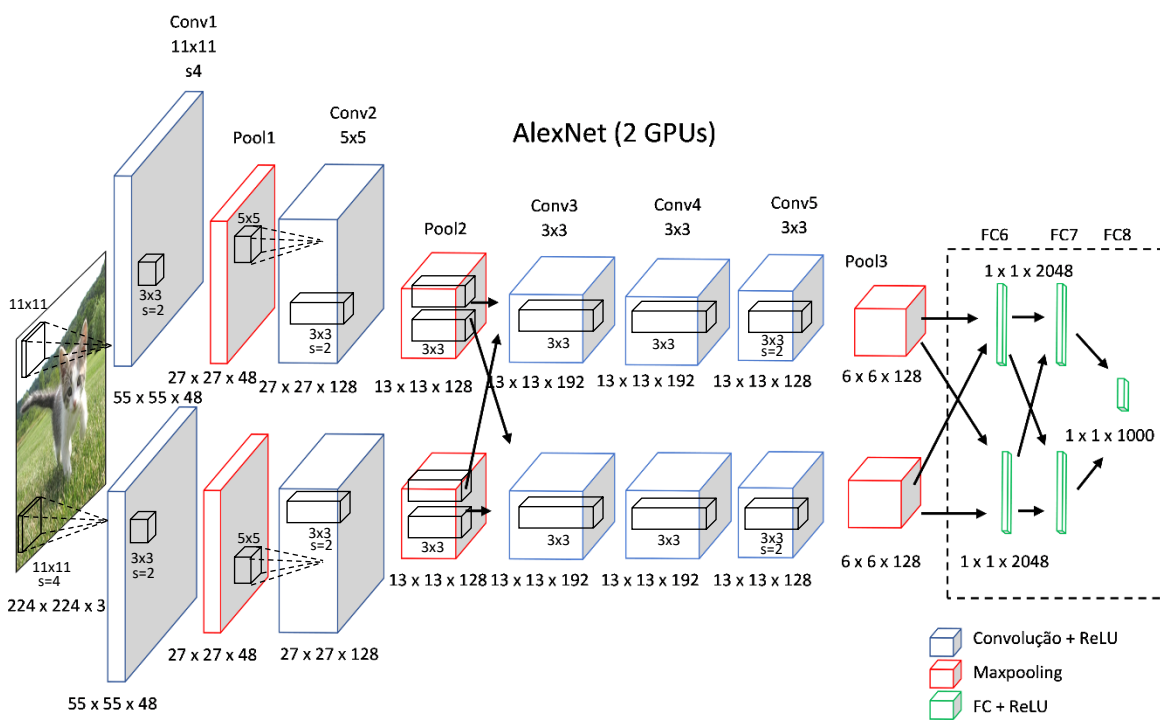


Figura 2.19 – Arquitetura de rede AlexNet. Adaptado de [ZHANG *et al.* 2020].

As imagens de resolução variável de ImageNet são reduzidas e recortadas com uma resolução fixa de  $256 \times 256$  pixels na entrada da rede. Na primeira camada de convolução, AlexNet usa campos receptivos grandes (*kernel*  $11 \times 11$  com passo 4) para diminuir o tamanho de dados de entrada, além das convoluções  $5 \times 5$  e  $3 \times 3$  com passo 1 nas camadas restantes. Ao contrário de LeNet-5, AlexNet usa a função de ativação linear ReLU, que treina mais rápido do que redes equivalentes cujos neurônios usam a função não linear saturante *tanh*. Além disso, AlexNet usa três camadas de subamostragem com *maxpooling*  $3 \times 3$  que se sobrepõem com passo 2, em vez de *average-pooling*  $2 \times 2$  não sobreposta com passo 2 de LeNet-5 [KRIZHEVSKY *et al.* 2012].

Após a última camada convolucional, AlexNet tem duas camadas totalmente conectadas com 4.096 saídas e uma camada de saída *softmax* totalmente conectada de 1.000 vias, que produz uma distribuição de probabilidade sobre os 1.000 rótulos de classe do subconjunto ImageNet, de forma que a soma das probabilidades seja 1 [RUSSAKOVSKY *et al.* 2015].

Devido à memória limitada nas primeiras GPUs, a rede AlexNet original foi dividida e treinada em duas GPUs (Figura 2.20), usando o algoritmo gradiente descendente estocástico com tamanho de lote (*mini-batch*) de 128 imagens.

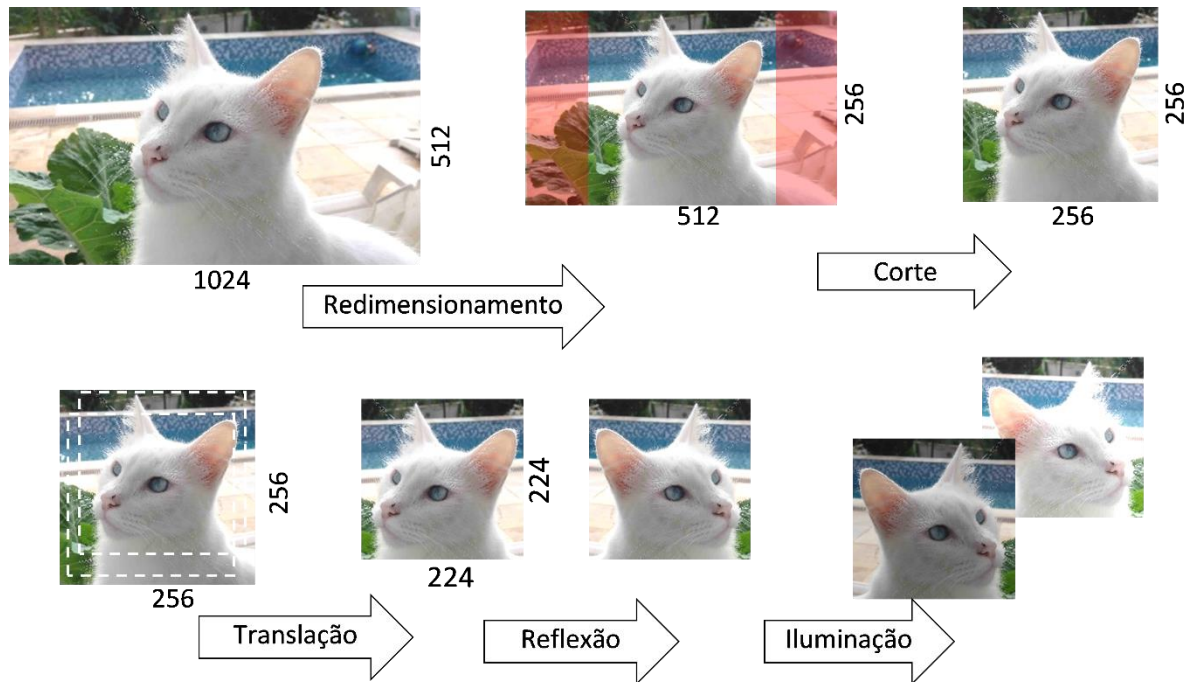


**Figura 2.20** – Mapas de AlexNet distribuídos em duas GPUs. Adaptado de [KRIZHEVSKY *et al.* 2012].

O tamanho e a complexidade da rede AlexNet com 60 milhões de parâmetros tornou o sobreajuste um problema significativo, sendo que a maior quantidade de parâmetros se encontra nas duas primeiras camadas totalmente conectadas com mais de 54 milhões de parâmetros. Sendo assim, o método de regularização *dropout* foi empregado nas duas primeiras camadas totalmente conectadas para reduzir o *overfitting*, zerando a saída de cada neurônio oculto com probabilidade de 0,5.

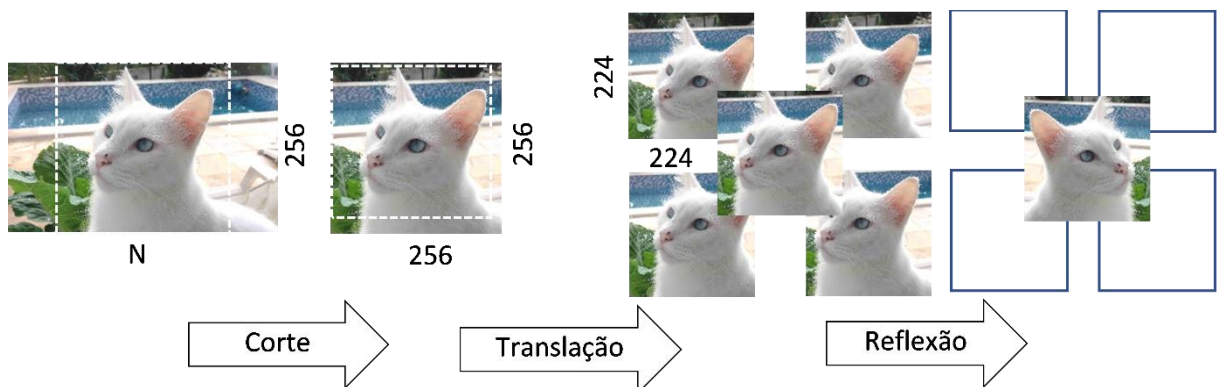


Outro método simples de regularização para reduzir o sobreajuste consiste em aumentar artificialmente o conjunto de dados de treinamento, usando transformações nas imagens originais com preservação dos rótulos, de forma que as imagens transformadas não precisem ser armazenadas em disco. A rede AlexNet aumenta artificialmente o conjunto de dados de treinamento usando três tipos de transformações de imagens: translação com corte aleatório, reflexão horizontal e alterações de iluminação/cor, como mostra a [Figura 2.21](#).



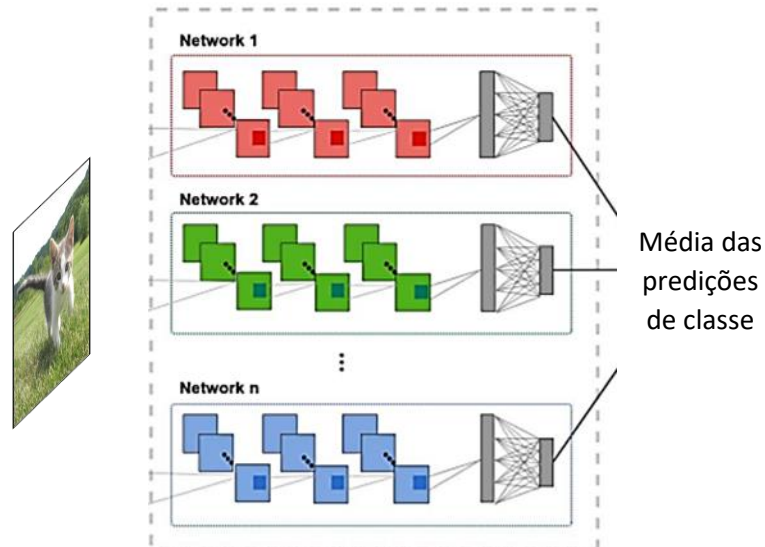
**Figura 2.21** – Aumento artificial do conjunto de dados de treinamento com transformações de imagens (redimensionamento e corte para  $256 \times 256$ , translação com corte aleatório de  $224 \times 224$ , reflexão horizontal e alterações de iluminação/cor).

Além disso, AlexNet combina as previsões de 10 transformações de imagem de teste em uma previsão final. Foram usados o recorte central e os quatro cantos, como também seus reflexos horizontais, como mostra a [Figura 2.22](#) [HOWARD 2014].



**Figura 2.22** – Previsão final pela combinação das previsões de 10 transformações da imagem de teste, usando o recorte central e os quatro cantos, bem como seus reflexos horizontais.

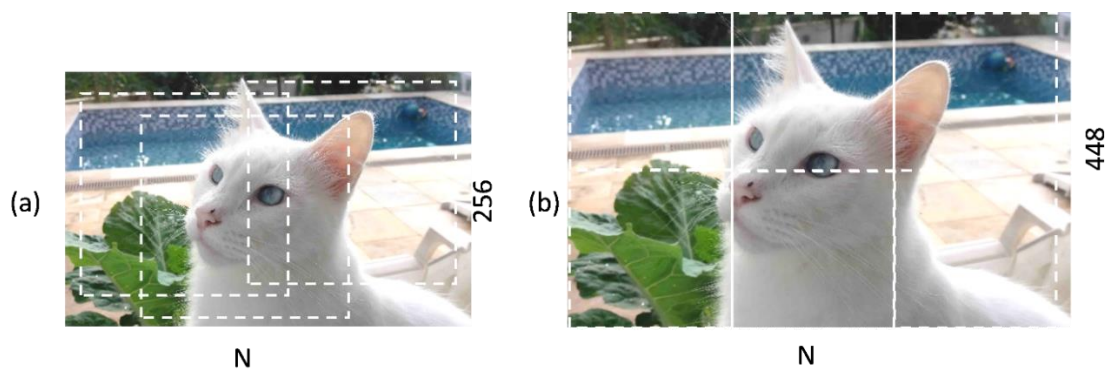
AlexNet demonstrou que sistemas baseados em rede neurais convolucionais superavam outras metodologias de aprendizado de máquina tradicionais concorrentes. A rede AlexNet vencedora da competição ILSVRC-2012 [DENG *et al.* 2012] era composta por uma montagem de sete redes profundas (Figura 2.23). A média de um conjunto de redes treinadas independentemente é uma solução eficaz para melhorar a precisão, amplamente adotada em competições de reconhecimento de imagens.



**Figura 2.23** – Rede composta de um conjunto de redes treinadas independentemente.

### 2.3.1.1 Melhorias na rede AlexNet

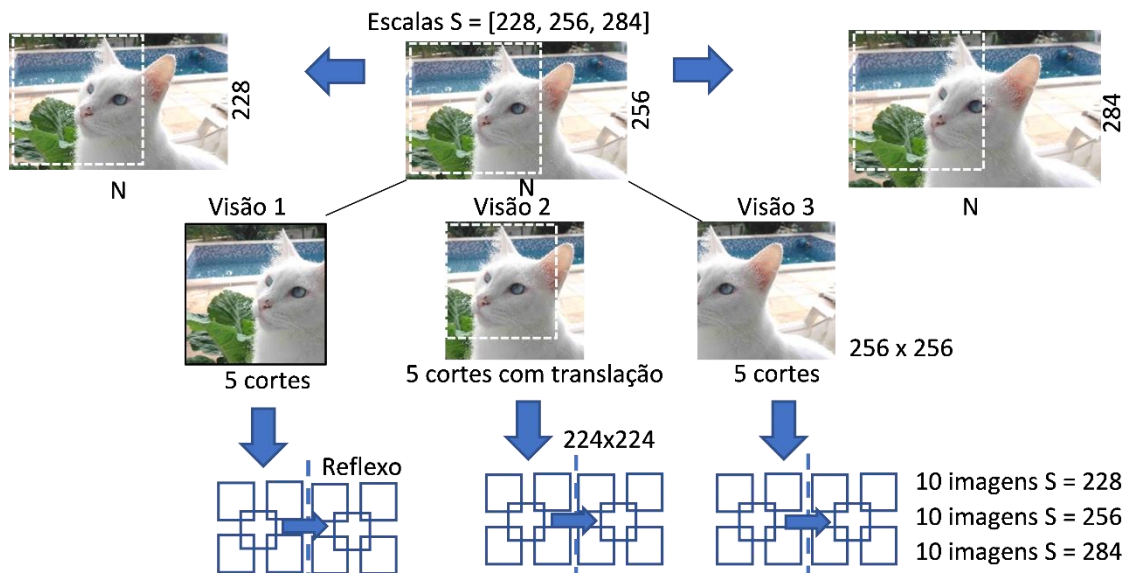
AlexNet incorporou o aumento do conjunto de dados de treinamento, transformações de dados no momento do teste e *dropout* para evitar ajustes excessivos. Posteriormente, Howard [HOWARD 2014] incorporou transformações adicionais usadas para aumentar o número efetivo de exemplos de treinamento e mais transformações usadas no momento do teste para melhorar a previsão. As transformações adicionais estendem a invariância de translação e invariância de cor com cortes aleatórios de  $224 \times 224$  sobre uma imagem retangular (redimensionada de  $256 \times N$  ou  $N \times 256$ ), para produzir um maior número de amostras de treinamento e ajudar a rede a aprender invariância de translação mais extensa (Figura 2.24a).



**Figura 2.24** – Transformações adicionais de imagens de treinamento, com cortes aleatórios de  $224 \times 224$  sobre uma imagem retangular redimensionada (a); recortes de  $224 \times 224$  de alta resolução em imagens de tamanho  $448 \times N$  (ou  $N \times 448$ ) (b).

Para os modelos de alta resolução, foram utilizados recortes de  $224 \times 224$  em imagens de treinamento de tamanho  $448 \times N$  (ou  $N \times 448$ ) (Figura 2.24b). Na prática, as imagens de treinamento  $256 \times N$  ( $N \times 256$ ) são reutilizadas, recortando-se *patches* de tamanho  $128 \times 128$  e redimensionando para  $224 \times 224$ , simulando o uso de *patches* de  $224 \times 224$  em uma imagem  $448 \times N$  ( $N \times 448$ ) sem a necessidade de armazenamento de imagens em várias resoluções. Os modelos de baixa resolução previamente treinados são usados para inicializar modelos de alta resolução para reduzir o tempo de treinamento de 90 para 30 épocas.

Como os objetos nas imagens podem existir em diferentes escalas, Howard treinou a rede em escala fixa (*patches*  $224 \times 224$  recortados de imagens  $256 \times N$  ou  $N \times 256$ ), mas prevê versões da imagem de teste em três escalas diferentes (228, 256, 284) para melhorar a previsão do conjunto (Figura 2.25). No entanto, quando a imagem de teste é muito ampliada, a rede treinada em uma escala fixa não tem um bom desempenho, sendo necessário treinar a rede novamente nesta escala para classificar objetos dentro de imagens ampliadas de alta resolução.



**Figura 2.25** – Transformações nas imagens de teste com a combinação de 5 cortes com translações, 2 espelhamentos, 3 escalas e 3 vistas com total de 90 imagens para previsões.

### 2.3.2 Modelo VGG

Após o desenvolvimento da rede AlexNet em 2012, houve uma tendência de tornar a rede convolucional cada vez mais profunda, impulsionada pela competição anual ILSVRC [RUSSAKOVSKY *et al.* 2014]. Em 2014, o *Visual Geometry Group* (VGG), da Universidade de Oxford, desenvolveu a rede VGG [SIMONYAN e ZISSERMAN 2015] mais profunda que AlexNet. A arquitetura VGG não difere da rede convolucional clássica de LeNet, mas melhorou o desempenho de classificação aumentando a profundidade da rede. Para demonstrar que a profundidade da rede convolucional é importante para precisão da classificação de imagens, o grupo realizou uma avaliação de redes de profundidades crescentes, onde uma pilha de camadas convolucionais é seguida por três camadas totalmente conectadas (*fully connected* – FC), como mostra as seguintes configurações na Figura 2.26: VGG-11 com 11 camadas (8 convolucionais e 3 camadas FC); VGG-13 (10 convolucionais e 3 camadas FC); VGG-16 (13 convolucionais e 3 camadas FC) e VGG-19 (16 convolucionais e 3 camadas FC) [SIMONYAN e ZISSERMAN 2015].



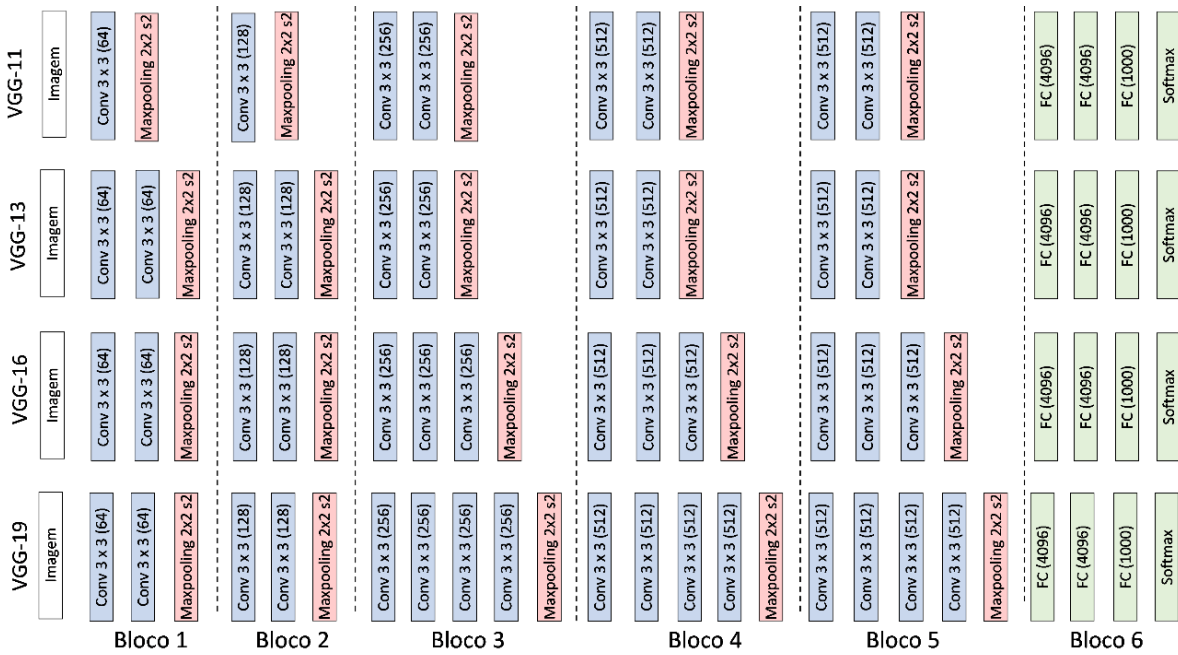


Figura 2.26 – Configurações VGG com 11 a 19 camadas [SIMONYAN e ZISSERMAN 2015].

Todas as redes VGG usam cinco blocos para facilitar a implementação de redes mais profundas. Um bloco consiste em uma sequência de camadas convolucionais, cada uma com preenchimento ou *same-padding* (para manter a resolução) e função de ativação ReLU, seguida por uma camada de *maxpooling*  $2 \times 2$  com passo 2 para redução da resolução espacial pela metade no final do bloco.

Em vez de usar *kernels* grandes nas primeiras camadas convolucionais (por exemplo, filtros  $5 \times 5$  de LeNet-5 ou filtros  $11 \times 11$  com passo 4 de AlexNet), foram empregados pequenos filtros  $3 \times 3$  com passo 1 em todas as camadas convolucionais. A Figura 2.27 mostra que duas camadas convolucionais  $3 \times 3$  consecutivas com passo 1 (sem agrupamento espacial) têm um campo receptivo efetivo de  $5 \times 5$ , e três camadas  $3 \times 3$  têm um campo de  $7 \times 7$ , com a vantagem de ter menos parâmetros e menor complexidade computacional [SIMONYAN e ZISSERMAN 2015].

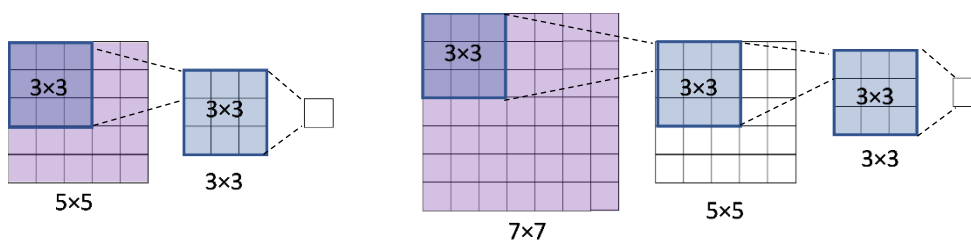
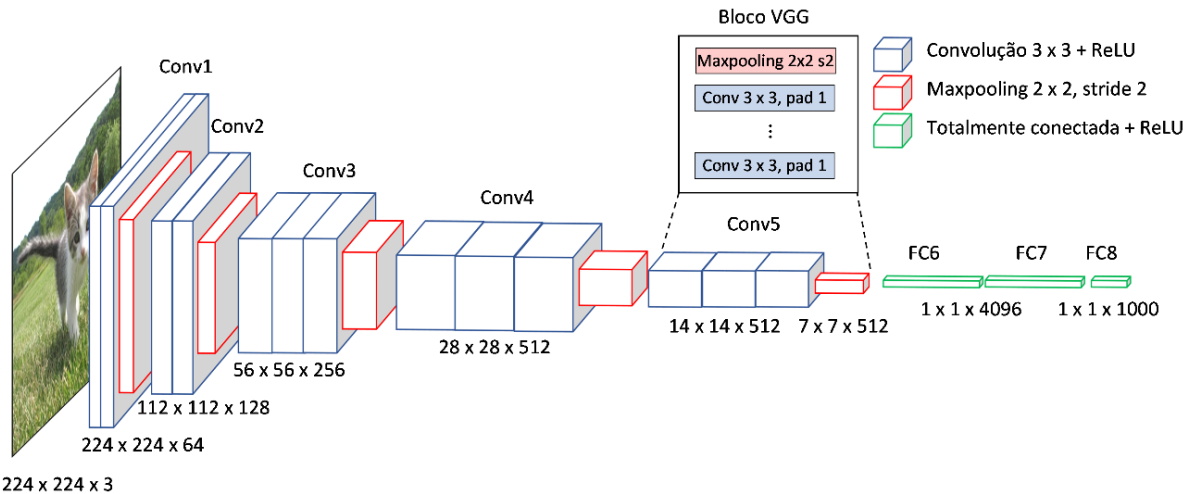


Figura 2.27 – Convoluções  $3 \times 3$  consecutivas com campo receptivo  $5 \times 5$  e  $7 \times 7$ .

A largura das camadas convolucionais (número de canais) é pequeno, começando com 64 canais no primeiro bloco e aumentando por um fator de 2 após cada camada *maxpooling*, até atingir 512 canais no último bloco [ZHANG *et al.* 2020]. Diferentes modelos VGG podem ser definidos por diferentes números de camadas convolucionais em cada bloco, como mostra a Figura 2.26. A Figura 2.28 mostra os mapas de atributos da rede VGG-16, sendo que cada bloco tem [2, 2, 3, 3, 3] camadas convolucionais com filtros  $3 \times 3$ .



**Figura 2.28** – Mapas de atributos da rede VGG-16 com 16 camadas (exceto maxpooling). Adaptado de [SIMONYAN e ZISSERMAN 2015].

Da mesma forma que AlexNet e LeNet-5, a rede VGG tem duas partes: a primeira composta de camadas convolucionais e subamostragem para extração de atributos e a segunda com camadas totalmente conectadas para classificação dos atributos. Todas as redes VGG têm a mesma configuração de camadas FC da rede AlexNet, onde as duas primeiras camadas FC têm 4.096 canais cada, e a última camada FC (*softmax*) realiza a classificação ILSVRC de 1.000 vias e, portanto, produz uma distribuição de probabilidade para 1.000 rótulos de classe do subconjunto ImageNet.

As redes VGG-16 e VGG-19 têm 138 e 144 milhões de parâmetros, respectivamente, sendo a maior parte nas camadas FC. Para evitar sobreajuste de dados, a rede usa regularização *dropout* com taxa de 0,5 nas duas primeiras camadas totalmente conectadas.

A entrada da rede é uma imagem RGB de tamanho fixo  $224 \times 224$ . No treinamento em escala única, uma imagem é redimensionada com o menor lado igual a 256 ou 384 e, em seguida, cortada aleatoriamente com tamanho  $224 \times 224$  (Figura 2.24a). Da mesma forma que AlexNet, é feito um aumento artificial de dados do conjunto de treinamento com reflexão horizontal, alteração de iluminação e cor aleatória. Porém, diferentemente de AlexNet, no treinamento em múltiplas escalas, um único modelo é treinado para reconhecer objetos em uma variedade de escalas, onde cada imagem de treinamento é redimensionada aleatoriamente dentro de um intervalo de [256, 512] para aumentar o conjunto de treinamento com variação de escala [SIMONYAN e ZISSERMAN 2015].

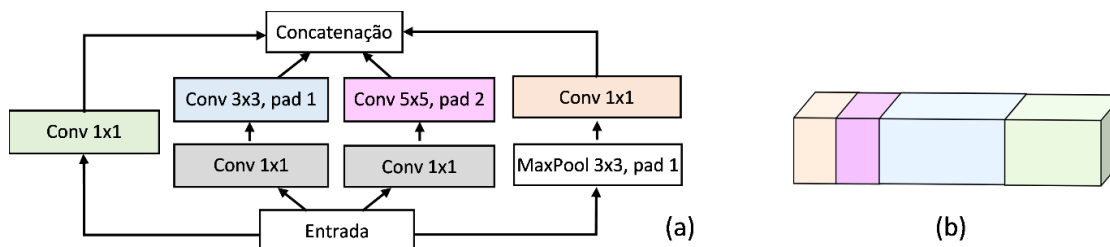
A rede VGG ficou em segundo lugar na tarefa de classificação do desafio ILSVRC-2014 com uma combinação de sete redes, atrás do vencedor GoogLeNet com 7 redes. Em termos de desempenho de rede única, a arquitetura VGG atinge o melhor resultado de classificação, superando o modelo único GoogLeNet [SIMONYAN e ZISSERMAN 2015].

Além da rede VGG, outras tentativas foram feitas para melhorar a arquitetura original de AlexNet (2012). Por exemplo, Clarifai (2013) [ZEILER e FERGUS 2014] e OverFeat (2014) [SERMANET *et al.* 2013] utilizaram um campo receptivo  $7 \times 7$  e passo 2 na primeira camada convolucional, obtendo um melhor desempenho no ILSVRC-2013. Outras melhorias foram obtidas treinando e testando as redes de forma densa em toda a imagem e em múltiplas escalas [SERMANET *et al.* 2013; HOWARD 2014].

### 2.3.3 Modelos Inceptions

A arquitetura Inception-v1 (2014) [SZEGEDY *et al.* 2015] com 22 camadas convolucionais é composta de blocos *Inception* com quatro caminhos paralelos de pequenos filtros de convolução, conforme ilustrado na Figura 2.29a. Os três primeiros caminhos usam camadas convolucionais com filtros  $1 \times 1$ ,  $3 \times 3$  e  $5 \times 5$  para extrair informações de diferentes tamanhos espaciais. Os dois caminhos do meio realizam primeiro uma convolução  $1 \times 1$  para reduzir o número de canais de entrada, reduzindo a complexidade do modelo. O quarto caminho usa uma camada de *maxpooling*  $3 \times 3$  para reduzir a resolução espacial, seguida por uma camada convolucional com filtro  $1 \times 1$  para reduzir o número de canais. Todos os quatro caminhos usam preenchimento apropriado para fornecer saídas com mesma altura e largura para concatenação. Finalmente, as saídas de cada caminho são concatenadas ao longo da dimensão do canal (Figura 2.29b). Assim, o bloco *Inception* da rede Inception-v1 concatena mapas de atributos produzidos por uma combinação de filtros de diferentes tamanhos, aumentando a largura da rede (número de canais). A combinação dos filtros explora a imagem em janelas variadas, capturando detalhes em diferentes escalas e agregando as informações para que o próximo estágio possa abstrair atributos de diferentes escalas simultaneamente [ZHANG *et al.* 2020].

Na Figura 2.29a, as convoluções  $1 \times 1$  (caixas cinzas) aplicam reduções de número de canais antes das convoluções  $3 \times 3$  e  $5 \times 5$ , e a convolução  $1 \times 1$  (caixa laranja) faz a projeção para redução de canais após o *maxpooling*. Sem estes redutores, a concatenação da saída do *maxpooling* (cujo número de filtros de saída é igual ao número de filtros na entrada) com as saídas das camadas convolucionais levaria a um aumento no número de mapas de atributos de cada bloco, com uma explosão da complexidade computacional em poucos estágios.

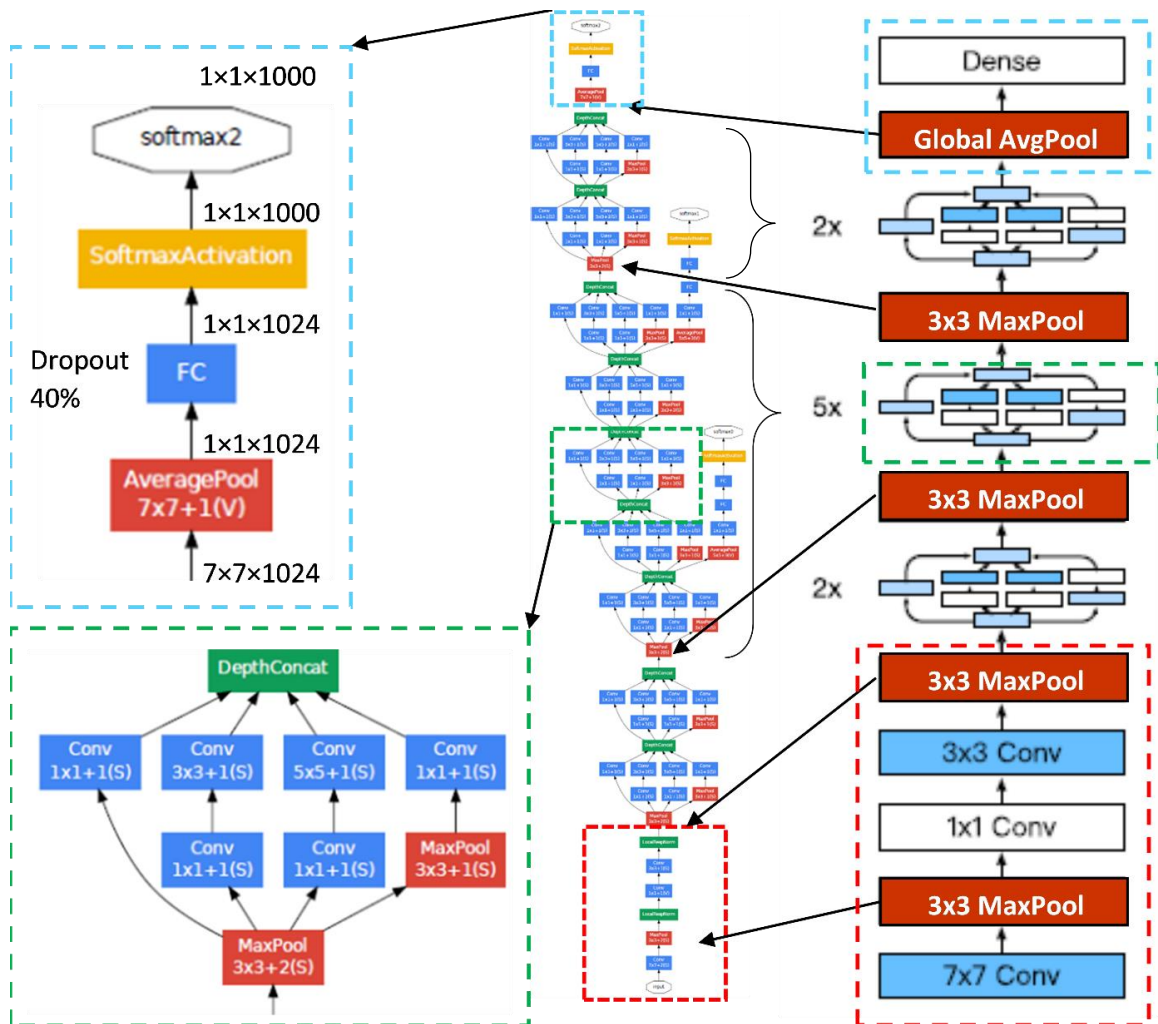


**Figura 2.29** – Estrutura paralela do bloco Inception-v1 (a) e mapas de atributos em diferentes escalas concatenados na saída do bloco (b). Adaptado de [ZHANG *et al.* 2020].

Como mostrado na Figura 2.30, uma rede *Inception* é composta por três pilhas de [2, 5, 2] blocos *Inception* com redutores de dimensão de canais (para evitar uma explosão descontrolada na complexidade computacional), intercaladas com *maxpooling* com passo 2 (para reduzir pela metade a resolução espacial). Todas as convoluções, incluindo as camadas de redução/projeção dentro dos módulos *Inception*, usam ativação linear retificada (ReLU).

No final, uma camada de agrupamento médio global (GAP – *Global Average Pooling*) evita uma pilha de três camadas totalmente conectada usada nas redes anteriores. Esta camada calcula a média de cada mapa de atributos de  $7 \times 7$  para  $1 \times 1$ . A próxima camada totalmente conectada (FC) permite adaptar e ajustar as redes com *fine-tuning* para outros conjuntos de rótulos. O uso de *dropout* permanece essencial na camada totalmente conectada. Um classificador *softmax* de 1.000 vias realiza a classificação.

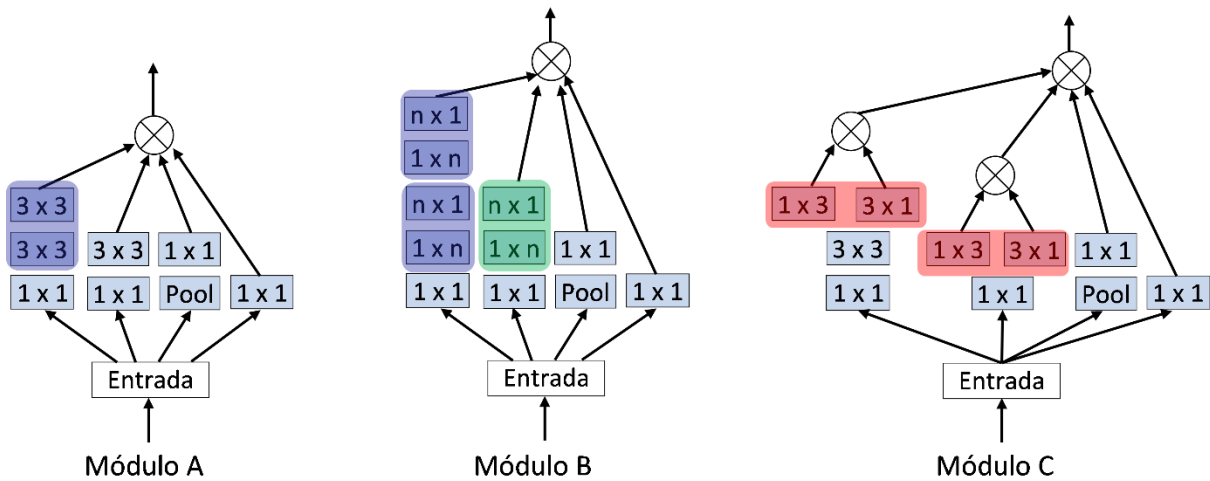
Apesar do modelo Inception-v1 ser computacionalmente complexo, com 6,8 milhões de parâmetros (12 vezes menos do que a arquitetura AlexNet), ele é significativamente mais preciso. Uma rede chamada GoogLeNet, com 7 versões do mesmo modelo *Inception* treinados independentemente, venceu o desafio ILSVRC-2014 em 1º lugar na tarefa de classificação, com a previsão dada pela média da classificação do conjunto das 7 redes.



**Figura 2.30** – Arquitetura de rede GoogLeNet com módulos Inception. Adaptado de [ZHANG *et al.* 2020; SZEGEDY *et al.* 2015].

Todos os modelos de estilo Inception são construídos como pilhas de módulos (blocos) Inception. Esta é a principal diferença das redes anteriores no estilo VGG, que eram pilhas de camadas de convolução simples. Embora os módulos Inception sejam extratores de atributos convolucionais semelhantes a convoluções, parecem ser capazes de aprender representações mais ricas com menos parâmetros.

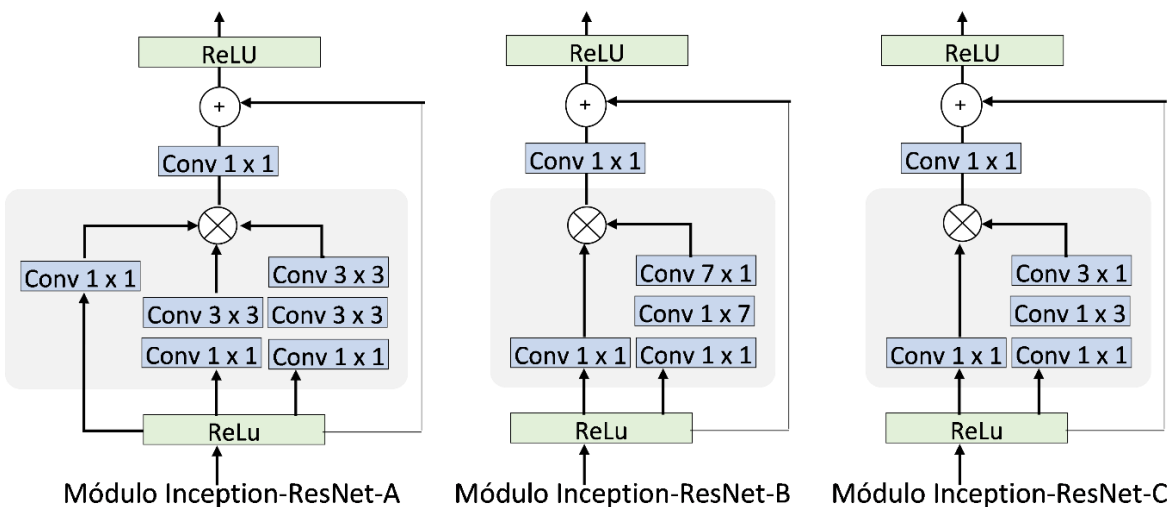
Várias versões de módulos Inception deram origem a outros modelos Inception mais refinados, como Inception-v2 e Inception-v3 [SZEGEDY *et al.* 2016] e Inception-v4 [SZEGEDY *et al.* 2017]. Estas redes usam módulos com convoluções fatoradas para aumentar a capacidade da rede Inception de maneira mais eficiente, sem aumentar os custos computacionais e número de parâmetros (Figura 2.31).



**Figura 2.31** – Outras versões de módulos Inception. Módulo A, onde cada convolução  $5 \times 5$  é fatorada em duas convoluções  $3 \times 3$ . Módulo B com a fatoração das convoluções  $n \times n$  do Módulo A em convoluções  $n \times 1$  e  $1 \times n$ . Módulo C, expansão da largura (número de canais) com convoluções  $n \times 1$  e  $1 \times n$  assimétricas paralelas [SZEGEDY et al. 2016].

A fatoração da convolução foi explorada de duas formas para aumentar a eficiência computacional: (1) fatoração em convoluções menores (por exemplo, uma convolução  $5 \times 5$  em duas convoluções  $3 \times 3$  ou uma convolução  $7 \times 7$  em três convoluções  $3 \times 3$ ) para reduzir o número de parâmetros (Figura 2.31a); (2) fatoração espacial em convoluções assimétricas (por exemplo, uma convolução  $n \times n$  em convoluções consecutivas ou paralelas  $1 \times n$  e  $n \times 1$ , com  $n = 3$  ou  $7$ ) para diminuir a complexidade (Figura 2.31b e c) [SZEGEDY et al. 2016].

Em 2016, a combinação de duas redes – ResNet (2015) com conexões residuais [HE et al. 2016a] e a última versão da arquitetura Inception-v3 (2015) [SZEGEDY et al. 2016] – deu origem à rede Inception-ResNet [SZEGEDY et al. 2017], que usa módulos Inception residuais com conexões de atalho (Figura 2.32). Como descrito na próxima seção, as redes residuais demonstraram que as conexões residuais aceleram o treinamento de redes muito profundas. Como as redes Inception tendem a ser profundas, o estágio de concatenação do filtro foi substituído por conexões residuais com adição de mapas de atributos.

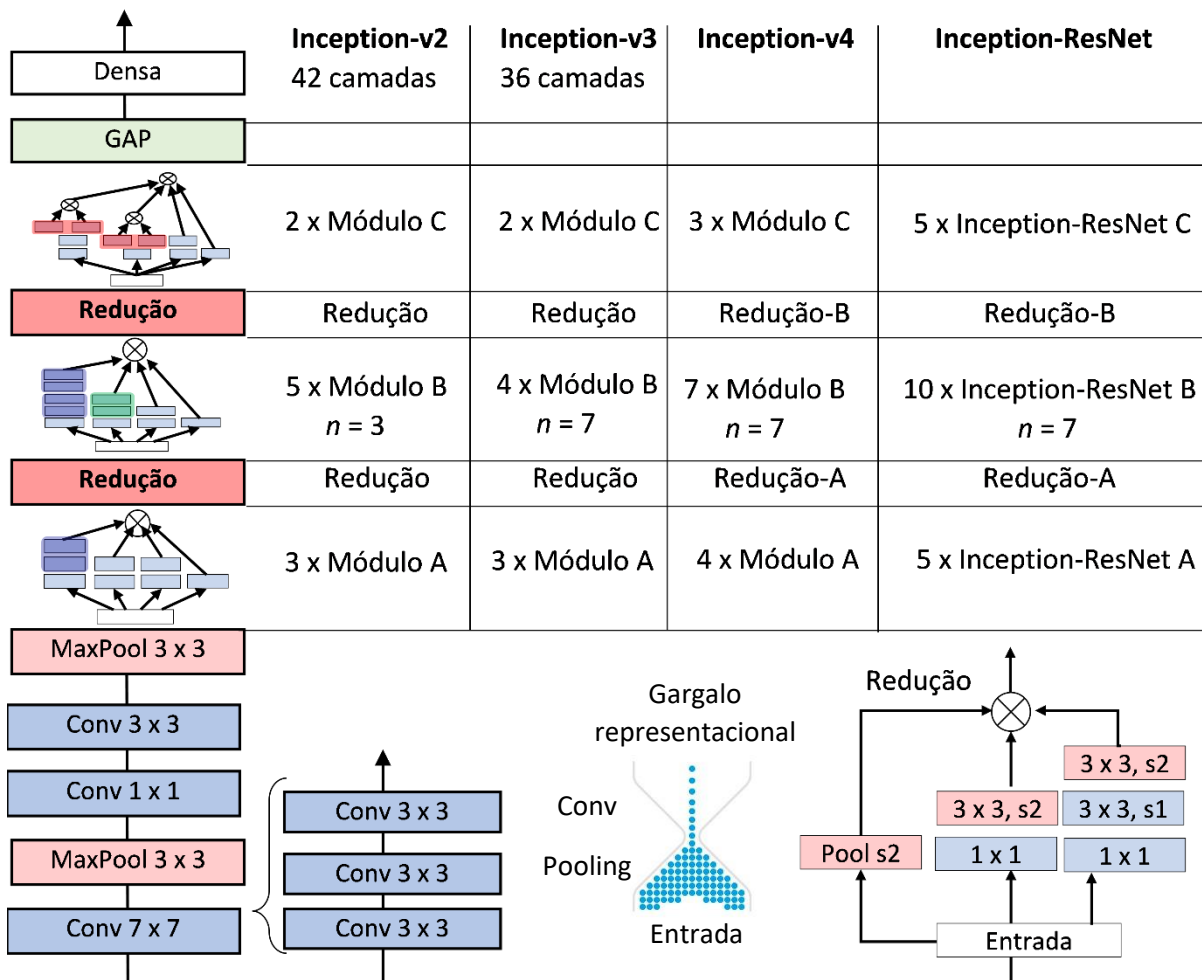


**Figura 2.32** – Módulos residuais Inception-ResNet A, B e C. Adaptado de [SZEGEDY et al. 2017].



Todas estas variantes seguem a mesma arquitetura básica da rede Inception-v1 original (Figura 2.30), substituindo-se os módulos Inception básico pelos módulos Inception A, B e C fatorados (Figura 2.31) ou módulos residuais Inception-ResNet A, B e C (Figura 2.32), conforme o esquema mostrado na Figura 2.33, onde as diferenças menos relevantes de cada modelo de rede foram omitidas.

Alguns princípios foram seguidos em todos os modelos de rede na Figura 2.33, como evitar gargalos representacionais que causam perda de informações devido à redução das dimensões da entrada antes de aumentar o número de canais. Em geral, o tamanho da representação deve diminuir suavemente das entradas para as saídas até a representação final. Para isso, estas redes substituem as camadas *maxpooling* da rede Inception-v1 original por um módulo redutor (Figura 2.33). Este módulo reduz o tamanho espacial da grade com uma operação de *pooling* no mapa de atributos de entrada, enquanto expande simultaneamente o número de canais pela concatenação de outros mapas de atributos após a convolução  $1 \times 1$  (com redução de canais para diminuir a complexidade) seguida de convolução  $3 \times 3$  com *stride* 1 (sem subamostragem espacial e com aumento do número de canais) e de convolução  $3 \times 3$  com *stride* 2 (com subamostragem) [SZEGEDY *et al.* 2016].



**Figura 2.33** – Esquema simplificado das redes Inception: substituição dos módulos Inception originais por módulos fatorados A, B e C da Figura 2.31 ou módulo Inception-ResNet A, B e C da Figura 2.32; substituição de *maxpooling* por módulos redutores de grade; e fatoração da convolução inicial  $7 \times 7$  em três convoluções  $3 \times 3$ . As variantes do módulo de redução A e B foram omitidas na figura. Baseado em [SZEGEDY *et al.* 2016; SZEGEDY *et al.* 2017].

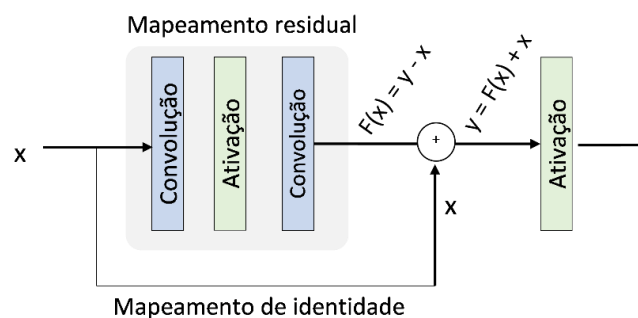
### 2.3.4 Modelo ResNet

Redes neurais representam atributos de baixo nível nas camadas iniciais até alto nível nas camadas finais, que podem ser enriquecidos pelo número de camadas empilhadas (profundidade da rede). Estudos anteriores revelaram a importância da profundidade da rede para obter melhor desempenho de classificação em grandes conjuntos de dados, com melhores resultados obtidos por modelos muito profundos, como VGG com 16 e 19 camadas [SIMONYAN e ZISSERMAN 2015], Inception-v1 com 22 camadas [SZEGEDY *et al.* 2015] e Inception-v3 com 36 camadas de profundidade [SZEGEDY *et al.* 2016].

Como contratempo, um problema de desaparecimento de gradientes dificultava a convergência do treinamento em redes muito profundas ao usar o algoritmo de aprendizado gradiente descendente estocástico (SGD) com *backpropagation*. Como o gradiente é retropropagado para as camadas anteriores, a multiplicação repetida de valores muito pequenos pode tornar o gradiente infinitamente pequeno nas camadas iniciais de redes muito profundas [GLOROT e BENGIO 2010].

Mesmo quando redes muito profundas convergem, pode ocorrer um problema de degradação do desempenho com saturação da precisão devido ao aumento da profundidade da rede. Esta degradação não é causada por *overfitting*, pois a adição de mais camadas a um modelo profundo leva a um maior erro de treinamento e, conseqüentemente, maior erro de teste, conforme experimentos em [HE *et al.* 2016a].

Para resolver este problema, foi definida uma estrutura de aprendizagem residual profunda, onde as camadas convolucionais empilhadas se ajustam a um mapeamento residual, em vez de se ajustar a uma função desejada. Na Figura 2.34, as camadas de convolução não lineares aproximam funções residuais  $F(x) = y - x$ . A função desejada torna-se  $y = F(x) + x$ , ou seja, a entrada somada ao resíduo  $F(x)$ , que pode ser realizada por redes neurais com conexões de atalho. Portanto, a conexão de atalho realiza o mapeamento de identidade  $x$  que é adicionada à saída do mapeamento residual  $F(x)$  das camadas convolucionais empilhadas.

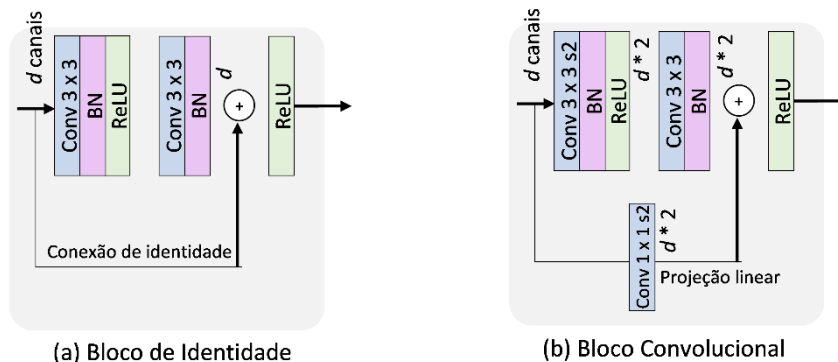


**Figura 2.34** – Bloco residual com conexão de atalho para mapeamento de identidade. Adaptado de [ZHANG *et al.* 2020].

Existem dois tipos de bloco residual na implementação da rede ResNet:

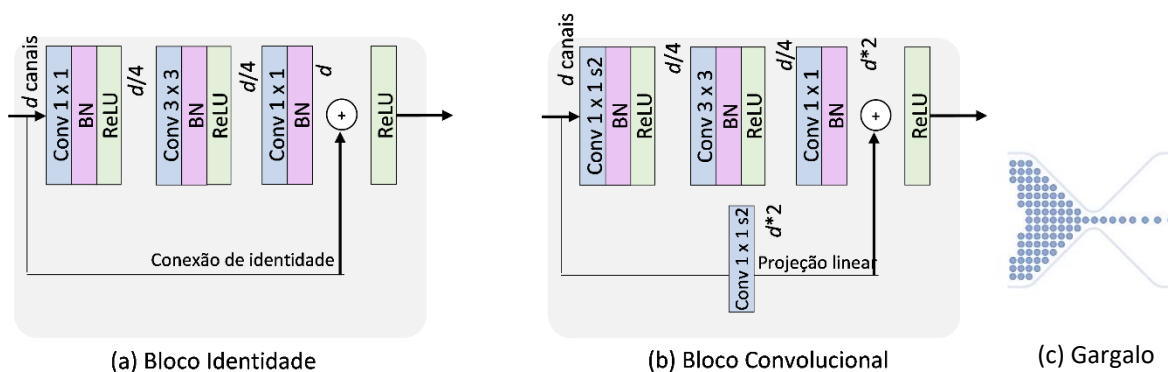
1) O bloco de identidade não possui camada de convolução no atalho. Na Figura 2.35a, as duas camadas convolucionais  $3 \times 3$  mantêm a resolução espacial e número de canais da entrada. A primeira camada convolucional é seguida por uma camada de normalização em lote (*Batch Normalization* – BN) [IOFFE e SZEGEDY 2015] e uma função de ativação ReLU. A segunda não linearidade ReLU é inserida após a adição. Em seguida, a conexão de atalho adiciona a entrada  $x$ , antes da função de ativação final ReLU.

2) O bloco convolucional possui uma camada de convolução no atalho. Na [Figura 2.35b](#), a camada convolucional  $1 \times 1$  com passo 2 no atalho realiza uma projeção linear da entrada para reduzir pela metade a resolução espacial e dobrar o número de canais da entrada, seguindo o mesmo formato da saída da pilha de camadas convolucionais  $3 \times 3$  para permitir a operação de adição.



**Figura 2.35** – *Blocos residuais regulares: bloco de identidade (a) que não altera a dimensão da entrada; bloco convolucional que altera a dimensão espacial da entrada (b). Adaptado de [ZHANG et al. 2020].*

Diferentemente dos blocos regulares da [Figura 2.35](#), que têm uma pilha de duas camadas de convoluções  $3 \times 3$ , os blocos de gargalo (*bottleneck*) da [Figura 2.36a e b](#) têm uma pilha de três camadas convolucionais  $1 \times 1$ ,  $3 \times 3$  e  $1 \times 1$ . Essas duas camadas de convolução  $1 \times 1$  reduzem e, em seguida, aumentam o número de canais da entrada, deixando a camada  $3 \times 3$  com um gargalo representacional ([Figura 2.36c](#)), diminuindo a complexidade e número de parâmetros da camada de convolução  $3 \times 3$ , sem degradar o desempenho, o que permite aumentar a profundidade da rede.



**Figura 2.36** – *Blocos residuais de gargalo: bloco de identidade (a); bloco convolucional (b).*

Uma arquitetura completa de rede residual ResNet foi desenvolvida em 2015 com o intuito de facilitar o treinamento de CNNs mais profundas. As diferentes configurações de redes ResNet na [Tabela 2.1](#) diferem em número de blocos residuais em cada módulo, com redes de 18 a 152 camadas convolucionais. As ResNets de 18 e 34 camadas usam blocos básicos regulares ([Figura 2.35](#)), mas as redes de 50, 101 e 152 camadas usam blocos de gargalo ([Figura 2.36](#)) [[ZHANG et al. 2020](#)].

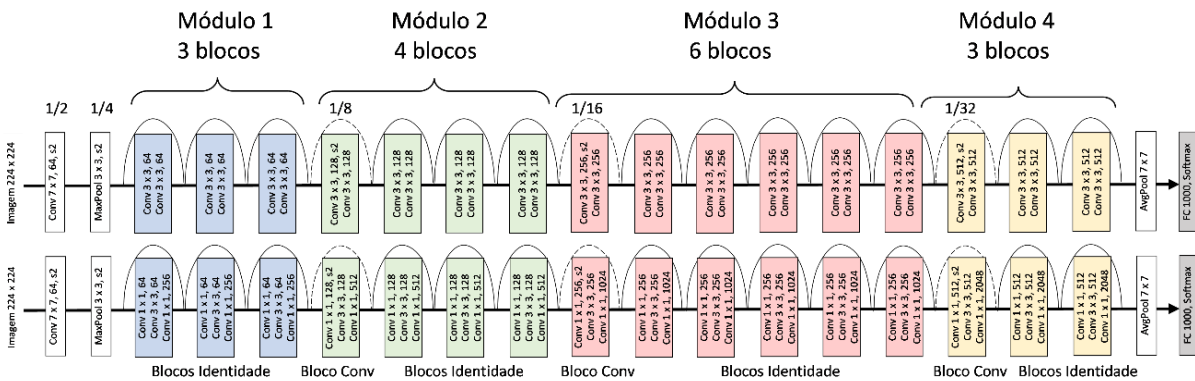


**Tabela 2.1** – *Arquiteturas ResNet. Os blocos residuais são mostrados entre colchetes. O número de blocos  $n$  é indicado ao lado do colchete para cada módulo conv- $m_x$ . A redução da resolução espacial é realizada pelo bloco residual convolucional em conv3\_1, conv4\_1 e conv5\_1 com passo 2 [HE et al. 2016a].*

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

As duas primeiras camadas da rede ResNet são as mesmas do GoogLeNet: uma camada convolucional  $7 \times 7$  com passo 2 e 64 canais de saída, seguida pela camada de *maxpooling*  $3 \times 3$  com passo 2 [ZHANG et al. 2020]. Em seguida, as redes ResNets apresentam quatro módulos residuais compostos de blocos de identidade e convolucionais. Cada módulo usa vários blocos residuais com o mesmo número de canais de saída. No primeiro módulo não é necessário reduzir a altura e a largura, uma vez que já foi utilizada uma camada de *maxpooling* com passo 2. No primeiro bloco residual de cada um dos módulos subsequentes, o número de canais é duplicado em relação ao módulo anterior, e a altura e a largura são reduzidas à metade. Por fim, assim como GoogLeNet, a rede termina com uma camada de agrupamento de média global, seguida por uma camada totalmente conectada *softmax* de 1.000 vias [RUSSAKOVSKY et al. 2015].

Na Figura 2.37 (inferior), a ResNet-50 tem [3, 4, 6, 3] x 3 camadas convolucionais em cada módulo (exceto a camada convolucional  $1 \times 1$  do atalho). Somando a primeira camada convolucional e a camada final totalmente conectada, existem 50 camadas no total. Os atalhos (arcos em linha cheia) nos blocos identidade são conexões de identidade (Figura 2.36a), e os arcos tracejados nos blocos convolucionais aumentam as dimensões de canais e diminuem a resolução espacial com projeções lineares (Figura 2.36b).



**Figura 2.37** – *Arquiteturas de ResNet-34 (topo) e ResNet-50 (inferior) [HE et al. 2016a].*

Comparando uma rede residual a uma rede simples equivalente sem conexões de atalho (com o mesmo número de parâmetros, profundidade, largura e custo computacional), He *et al.* demonstraram que as redes residuais profundas são mais fáceis de otimizar [HE *et al.* 2016a]. As redes simples correspondentes apresentam maior erro de treinamento quando a profundidade aumenta e, conseqüentemente, ocorre degradação do desempenho no conjunto de teste. Eles também concluíram que as redes residuais profundas apresentam ganhos de precisão com o aumento da profundidade, produzindo resultados melhores do que as redes menos profundas anteriores, como VGG e GoogLeNet. Um conjunto de seis redes residuais ResNet-v1 de diferentes profundidades conquistou o 1º lugar no desafio de classificação ILSVRC-2015 [RUSSAKOVSKY *et al.* 2015].

Posteriormente, outras arquiteturas também incorporaram o conceito de blocos residuais, como as redes ResNet in ResNet [TARG *et al.* 2016], WideResNet [ZAGORUYKO e KOMODAKIS 2016], ResNeXt [XIE *et al.* 2017], Inception-ResNet [SZEGEDY *et al.* 2017], incluindo a rede Xception [CHOLLET 2017] que substitui os módulos Inception por módulos de convoluções separáveis em profundidade com conexões residuais.

### Redes residuais WideResNet (WRN)

Com um aumento gradual da profundidade das redes AlexNet, VGG e diferentes versões de redes Inceptions, o treinamento de redes neurais mais profundas apresentaram algumas dificuldades, incluindo degradação da precisão e explosão/desaparecimento de gradientes. As redes ResNets adicionaram conexões residuais para amenizar este problema e permitir o treinamento de redes neurais mais profundas. Além disso, os autores das redes residuais tentaram torná-las o mais finas possível (diminuindo o número de canais) para aumentar a sua profundidade com menos parâmetros. Também introduziram um bloco de gargalo que diminui o número de canais com uma convolução  $1 \times 1$ , diminuindo a complexidade da camada convolucional  $3 \times 3$  seguinte. Entretanto, as redes WideResNets (WRNs) [ZAGORUYKO e KOMODAKIS 2016] – redes residuais largas ou amplas – diminuíram a profundidade e aumentaram a largura das redes residuais, superando suas contrapartes finas e profundas.

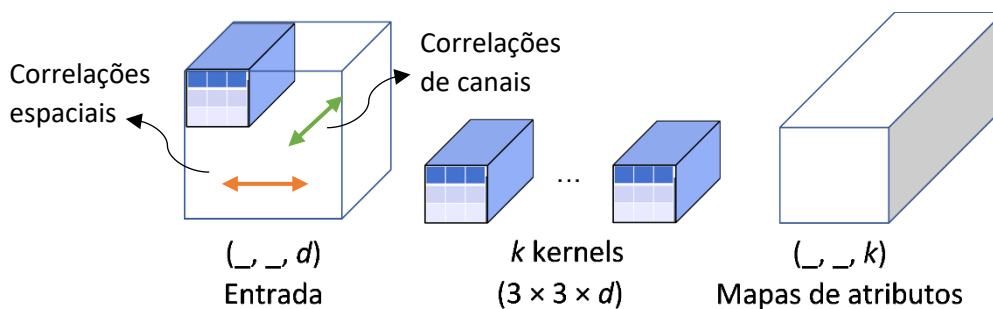
Para ImageNet, as redes ResNet-18 e ResNet-34 da Tabela 2.1 foram treinadas usando blocos regulares de duas camadas convolucionais  $3 \times 3$ , aumentando gradualmente sua largura com um fator  $k$  de 1 para 4. As redes residuais originais com  $k = 1$  são referidas como finas e as redes com  $k > 1$  como largas. A largura da rede nos três módulos Conv2-4 é escalada pelo fator  $k$ . Assim, a notação WRN- $n$ - $k$  denota uma rede residual que tem um número total de camadas convolucionais  $n$  e um fator de largura  $k$ . Foi demonstrado que o aumento da largura aumenta gradualmente a precisão de ambas as redes residuais [ZAGORUYKO e KOMODAKIS 2016].

A rede WRN-50-2, baseada na rede ResNet-50 da Figura 2.37 com blocos residuais de gargalo e fator de largura  $k = 2$ , supera a rede ResNet-152 com 3 vezes menos camadas, com um número comparável de parâmetros em torno de 66 milhões, sendo significativamente mais rápida de treinar. Assim, o alargamento da rede mostrou-se mais eficaz para melhorar o desempenho das redes residuais em comparação com o aumento de sua profundidade [ZAGORUYKO e KOMODAKIS 2016].

### 2.3.5 Modelo Xception

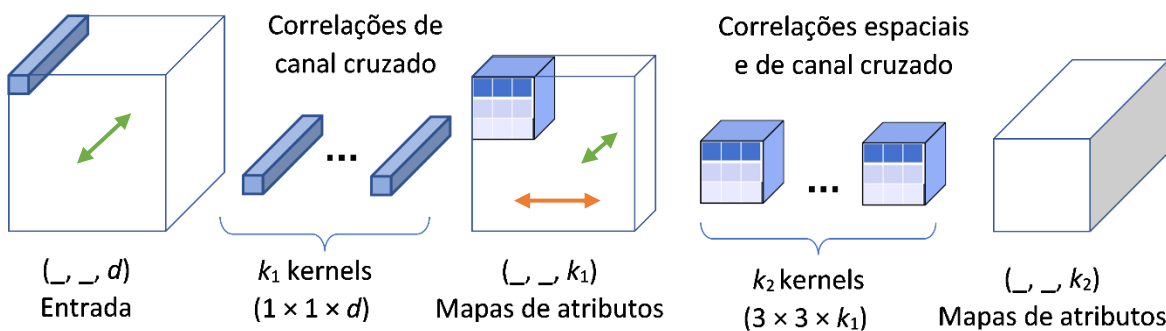
A arquitetura de rede neural Xception (*Extreme Inception*) [CHOLLET 2017] proposta em 2016 foi inspirada na rede Inception. Da mesma forma que Inception-ResNet, a rede Xception funde as ideias de GoogLeNet e ResNet, mas substitui os módulos Inception por módulos de convoluções separáveis em profundidade com conexões residuais.

Uma camada de convolução regular (Figura 2.38) aprende filtros 3D com duas dimensões espaciais (largura e altura) e uma dimensão de canal  $d$ . Assim, um único *kernel* de convolução captura simultaneamente correlações espaciais e correlações de canal, ou seja, padrões espaciais (atributos presentes espacialmente em cada um dos canais) e padrões de canais cruzados (atributos locais presentes em todos os canais).



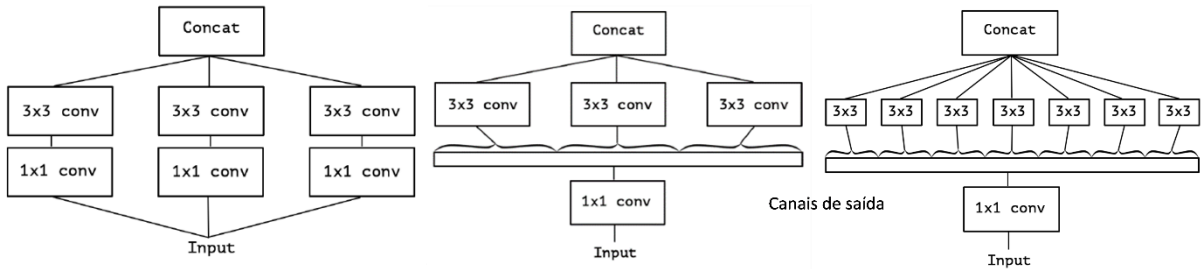
**Figura 2.38** – Convolução regular captura correlações espaciais e correlações entre canais simultaneamente.

O módulo Inception contém alguns ramos que primeiro analisam as correlações de canal cruzado da entrada por meio das convoluções  $1 \times 1$  e mapeiam os dados de entrada em 3 ou 4 espaços separados que têm menos canais do que o espaço de entrada original. Depois, todas as correlações (espaciais e de canal cruzado) nesses espaços 3D menores são mapeadas por meio de convoluções regulares  $3 \times 3$  ou  $5 \times 5$ , como mostrado na Figura 2.29. Assim, cada ramo de camadas convolucionais capturam padrões de canal cruzado com filtros  $1 \times 1$  e, em seguida, a camada de convolução regular  $n \times n$  procura padrões espaciais e de canais cruzados em conjunto, como mostra a Figura 2.39.

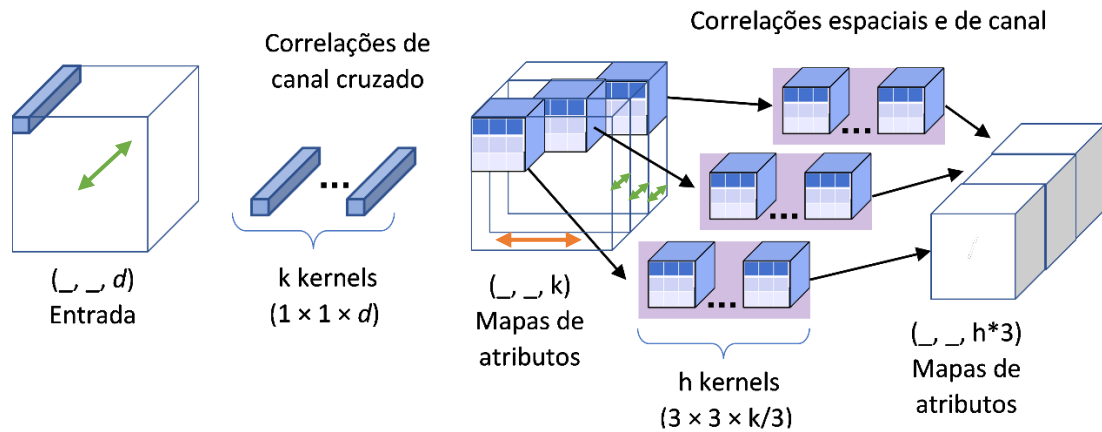


**Figura 2.39** – Convoluções  $1 \times 1$  seguidas de convoluções regulares usada em um ramo do módulo Inception original.

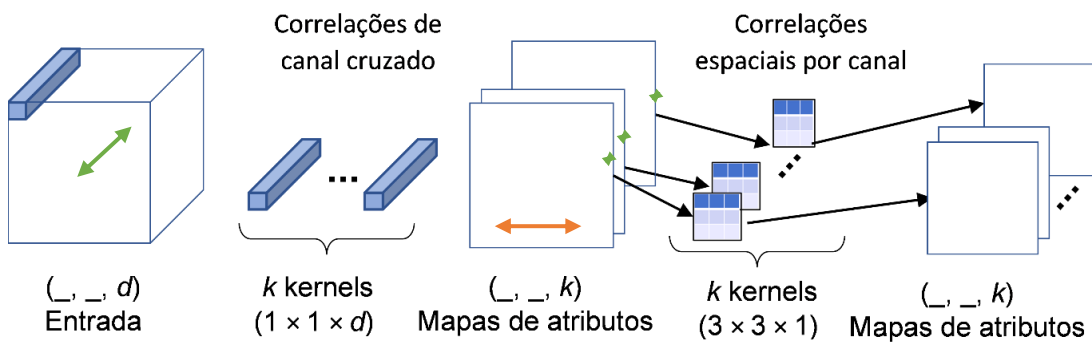
Considerando uma versão simplificada do módulo Inception, que contém apenas convoluções  $3 \times 3$ , sem nenhum ramo de subamostragem (*pooling*), como mostrado na Figura 2.40a, o módulo pode ser reformulado como uma grande convolução  $1 \times 1$ , seguida por convoluções regulares que operam em segmentos não sobrepostos dos canais de saída (Figura 2.40b e Figura 2.41). Aumentando hipoteticamente o número de segmentos ao extremo, uma versão extrema de um módulo Inception usaria uma convolução  $1 \times 1$  para mapear correlações entre canais e, então, uma convolução espacial para mapear as correlações espaciais de cada canal de saída separadamente (Figura 2.40c e Figura 2.42).



**Figura 2.40** – Módulo Inception original simplificado (a) e módulo Inception simplificado equivalente (b). Versão extrema do módulo Inception: convoluções de canal cruzado ( $1 \times 1$ ) e convoluções espaciais 2D ( $3 \times 3$ ) por canal [CHOLLET 2017].

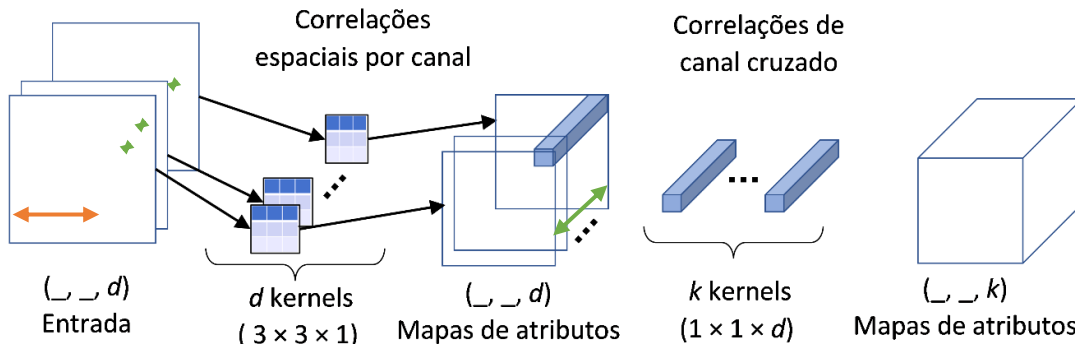


**Figura 2.41** – Convoluções do módulo Inception simplificado: convoluções de canal cruzado ( $1 \times 1$ ) e convoluções regulares 3D ( $3 \times 3$ ).



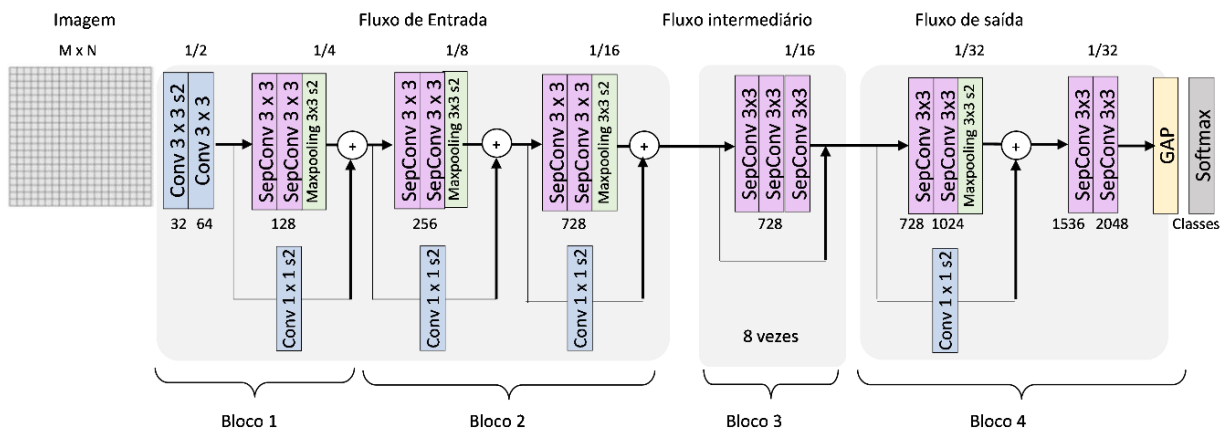
**Figura 2.42** – Versão extrema do módulo Inception: convoluções de canal cruzado ( $1 \times 1$ ) e convoluções espaciais 2D ( $3 \times 3$ ) por canal.

A versão extrema do módulo Inception é parecida com uma convolução separável em profundidade, também chamada de convolução separável (DepthSepConv) em frameworks de aprendizagem profunda, como TensorFlow [ABADI *et al.* 2015] e Keras [CHOLLET 2018]. A principal diferença é a ordem das operações, como mostram as Figuras 2.42 e 2.43. Em outras palavras, a convolução separável (Figura 2.43) consiste primeiro em uma convolução em profundidade, ou seja, uma convolução espacial com um único filtro 2D ( $n \times n$ ) realizada independentemente em cada canal de entrada. Em seguida, uma convolução pontual, ou seja, uma única camada de convolução com filtros 3D ( $1 \times 1 \times d$ ) procura padrões de canal cruzado, projetando os canais de saída da convolução em profundidade em um novo espaço de canal [CHOLLET 2017].



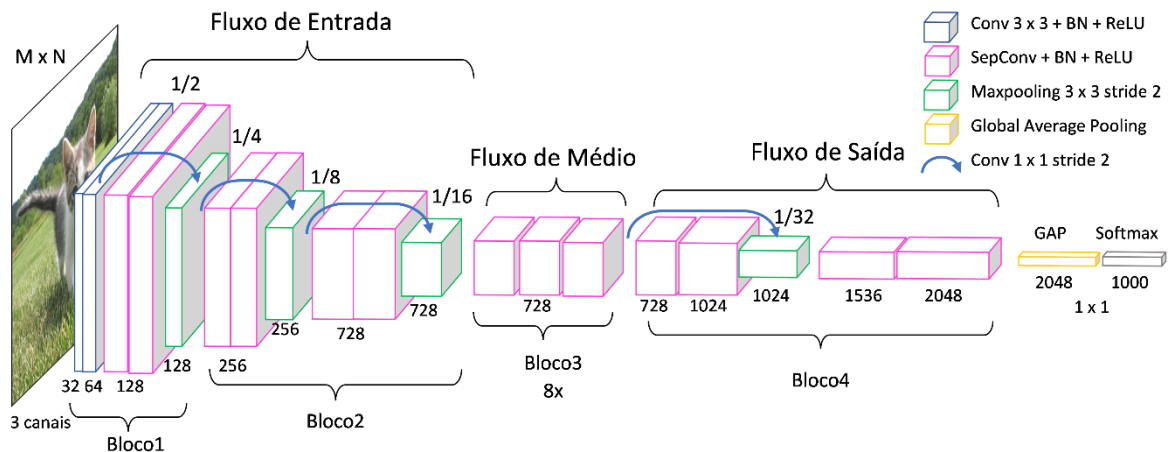
**Figura 2.43** – *Convoluções separáveis em profundidade: convoluções espaciais 2D (3 × 3) por canal e convoluções de canal cruzado (1 × 1).*

Com estas considerações, em 2017, Chollet propôs a substituição dos módulos Inception por convoluções separáveis em profundidade para criar uma nova arquitetura Xception (*Extreme Inception*) [CHOLLET 2017]. Nesta arquitetura, os dados passam pelo fluxo de entrada, pelo fluxo intermediário repetido oito vezes e, finalmente, pelo fluxo de saída (Figura 2.44). A rede possui 36 camadas convolucionais para extração de atributos, ou seja, duas camadas convolucionais regulares no início da rede, e, o restante, convoluções separáveis com algumas camadas de *maxpooling* intermediárias. Essas camadas são estruturadas em 14 módulos com conexões residuais, exceto o primeiro e o último módulo. As camadas finais usam um agrupamento de média global (GAP) e uma camada densa *softmax* opcional de saída para classificação de 1.000 rótulos de ImageNet.



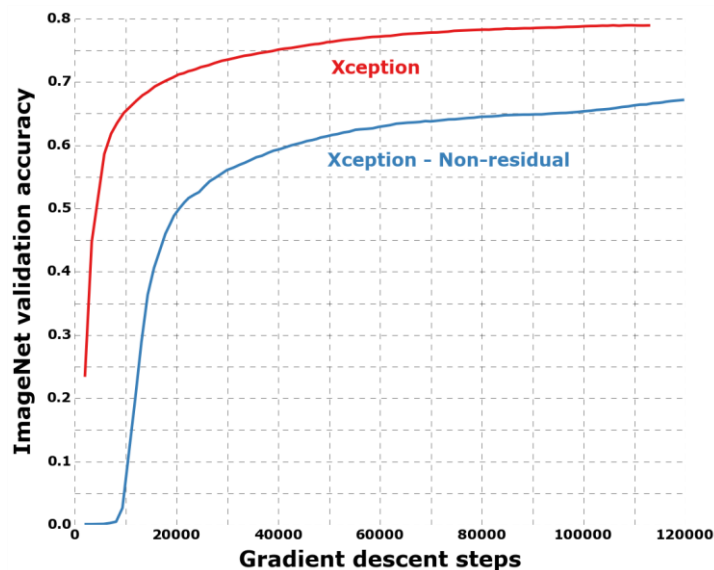
**Figura 2.44** – *Fluxo de entrada, intermediário e de saída da arquitetura Xception com módulos residuais de convoluções separáveis em profundidade. Baseado em [CHOLLET 2017].*

A [Figura 2.45](#) mostra alguns mapas de atributos da rede Xception.



**Figura 2.45** – Mapas de atributos da rede Xception.

Como vimos, as conexões residuais são essenciais para aceleração da convergência de modelos de redes ResNets profundos, assim como para obter melhorias de desempenho na classificação final. Para avaliar os benefícios das conexões residuais na arquitetura Xception, ela foi comparada com uma versão modificada do Xception sem conexão residual, obtendo maior desempenho [CHOLLET 2017], como mostram os resultados na [Figura 2.46](#).



**Figura 2.46** – Treinamento da rede Xception com / sem conexões residuais [CHOLLET 2017].

As conexões residuais não são necessárias para construir modelos com convoluções separáveis em profundidade, mas este resultado mostra a importância das conexões residuais para esta arquitetura específica. Modelos não residuais do estilo VGG, onde todas as camadas de convolução foram substituídas por convoluções separáveis em profundidade, também podem obter bons resultados. A operação de convolução separável em profundidade foi adotada em outras redes neurais recentes, como MobileNets-v1 e v2 [HOWARD *et al.* 2017; SANDLER *et al.* 2018] e ShuffleNet [ZHANG *et al.* 2017].



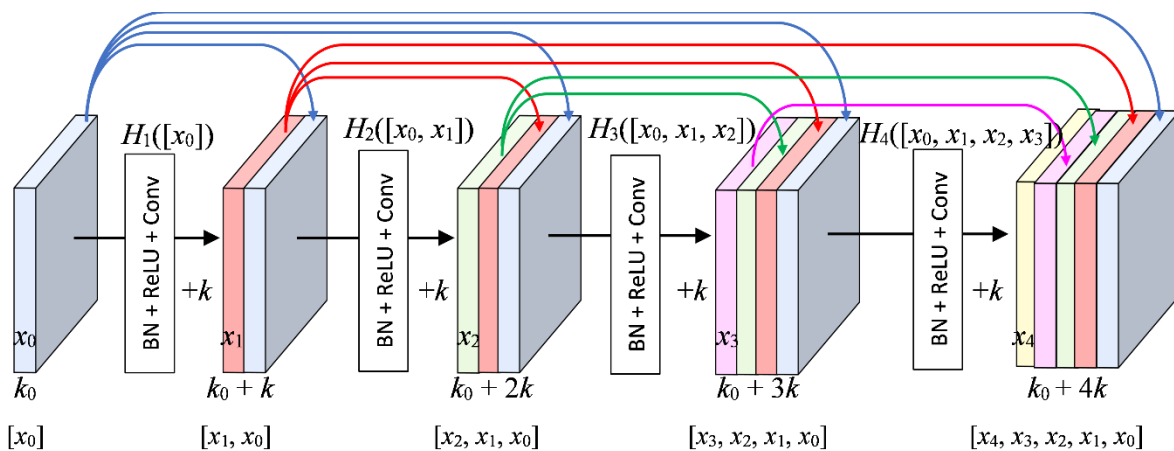
### 2.3.6 Modelo DenseNet

À medida que as CNNs se tornavam cada vez mais profundas, surgia um novo problema de desaparecimento de gradiente. As redes ResNets abordaram este problema desviando o sinal por meio de conexões de identidade, criando caminhos curtos das camadas iniciais para as camadas posteriores [HE *et al.* 2016a]. Com isso, foi mostrado que as redes neurais convolucionais podem ser mais profundas, mais precisas e eficientes de treinar, se contiverem conexões de atalho. As redes residuais também facilitaram o treinamento de redes com mais de 1.000 camadas em ResNet-v2 [HE *et al.* 2016b].

Em 2018, foi criada uma rede convolucional densa, chamada DenseNet [HUANG *et al.* 2017], que conecta cada camada a todas as outras em um modo *feedforward*. Para cada camada, os mapas de atributos de todas as camadas anteriores são usados como entradas, e seu próprio mapa de atributos de saída é usado como entrada de todas as camadas posteriores. Assim, a DenseNet alivia o problema do desaparecimento de gradiente, propaga e reutiliza atributos, além de reduzir o número de parâmetros, exigindo menos computação para alcançar alto desempenho.

A Figura 2.47 mostra um bloco denso composto de quatro blocos convolucionais (retângulos brancos), com um mapa de atributos de entrada (caixa azul) e mapas de atributos de saída dos blocos convolucionais (caixa vermelha, verde, rosa e amarela). Considere uma imagem  $x_0$  na entrada de um bloco denso, com  $L = 4$  blocos convolucionais que implementam uma função composta não linear  $H_l(\cdot)$ , onde  $l$  é o índice do bloco convolucional de 1 a  $L$ . Motivado por ResNet-v2 [HE *et al.* 2016b], a função  $H_l(\cdot)$  de um bloco convolucional é definida por três operações consecutivas: normalização em lote (BN) [IOFFE e SZEGEDY 2015], seguida por uma unidade linear retificada (ReLU) [NAIR e HINTON 2010] e uma convolução  $3 \times 3$  (Conv).

A saída do bloco convolucional  $l$  é denotada por  $x_l$ . Para melhorar o fluxo de informações entre os blocos, foram introduzidas conexões diretas de qualquer bloco para todos os blocos posteriores. Conseqüentemente, o bloco  $l$  recebe como entrada os mapas de atributos de todos os blocos anteriores, onde  $[x_0, x_1, \dots, x_{l-1}]$  refere-se à concatenação dos mapas de atributos produzidos nos blocos convolucionais  $0, \dots, l-1$ , que produz a saída  $x_l = H_l([x_0, \dots, x_{l-1}])$ .



**Figura 2.47** – Bloco denso de 4 blocos convolucionais, onde cada bloco usa todos os mapas de atributos anteriores como entrada. Taxa de crescimento de  $k$  mapas de atributos dentro de um bloco denso. Adaptado de [HUANG *et al.* 2017].

Em vez de aumentar a complexidade da rede com uma arquitetura extremamente profunda, como ResNet-v2 [HE *et al.* 2016b], ou mais larga, como WideResNet [ZAGORUYKO e KOMODAKIS 2016], a DenseNet explora a reutilização de atributos, produzindo modelos com menos parâmetros. Assim, diferente das ResNets que combinam atributos por somatório antes de serem passados ao próximo bloco convolucional, a DenseNet concatena os mapas de atributos dentro do bloco denso, como mostra a Figura 2.47. Consequentemente, um bloco convolucional tem como entrada os mapas de atributos de todos os blocos convolucionais anteriores, e cada bloco adiciona apenas um pequeno conjunto de  $k$  mapas de atributos (*taxa de crescimento*), mantendo os mapas anteriores inalterados. Em comparação com as redes Inception [SZEGEDY *et al.* 2015; SZEGEDY *et al.* 2016], que também concatenam mapas de atributos de diferentes camadas, as DenseNets são mais simples e eficientes [HUANG *et al.* 2017].

Na Figura 2.47, o bloco denso é composto de 4 blocos convolucionais DenseNet básico com uma convolução  $3 \times 3$  (Figura 2.48a). Assim como nas redes Inception [SZEGEDY *et al.* 2016] e ResNet [HE *et al.* 2016a], a técnica de gargalo foi adotada nos blocos convolucionais DenseNet-B (*bottleneck*) (Figura 2.48b), ou seja, uma convolução  $1 \times 1$  foi introduzida antes de cada convolução  $3 \times 3$ , para reduzir o número de mapas de atributos de entrada e melhorar a eficiência computacional.

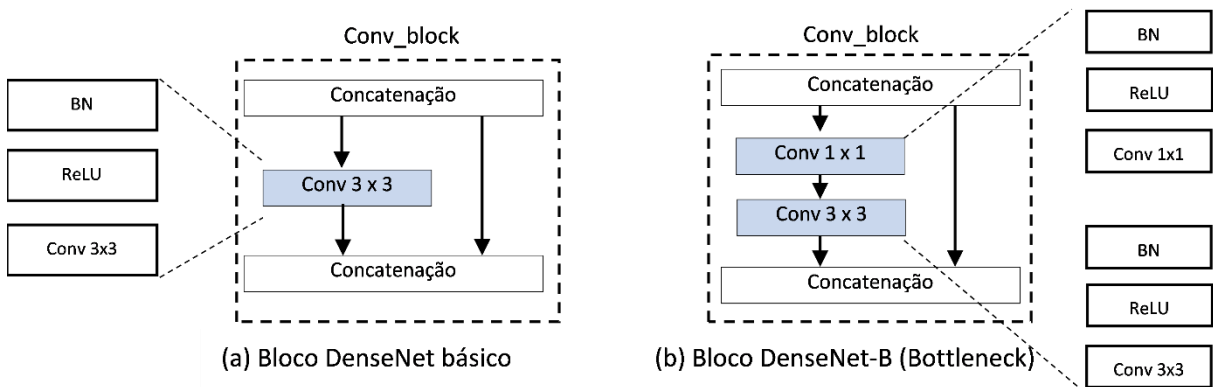


Figura 2.48 – Tipos de blocos convolucionais DenseNet.

A Figura 2.49 mostra uma rede DenseNet completa dividida em vários blocos densos, com camadas de transição (caixas verdes) que conectam dois blocos densos adjacentes. Como o tamanho dos mapas de atributos é mantido dentro do bloco denso, a camada de transição reduz as dimensões espaciais do mapa de atributos pela metade. As camadas de transição consistem em uma camada de normalização em lote (BN), uma função de ativação ReLU, uma camada convolucional  $1 \times 1$ , seguida por uma camada de agrupamento médio  $2 \times 2$  com passo 2 (*average-pooling*), como mostra a Figura 2.50.

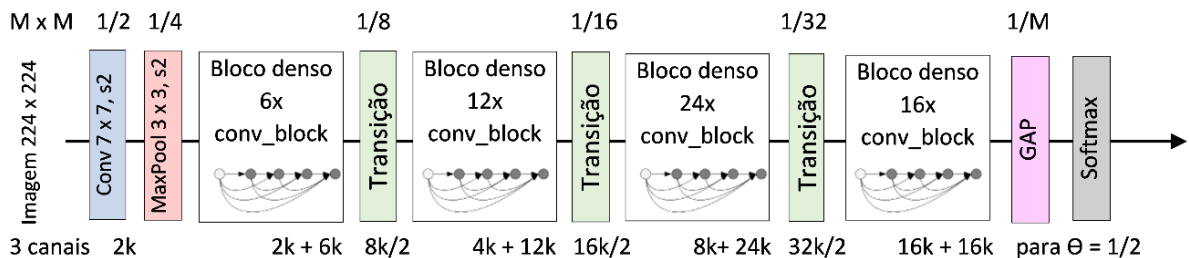
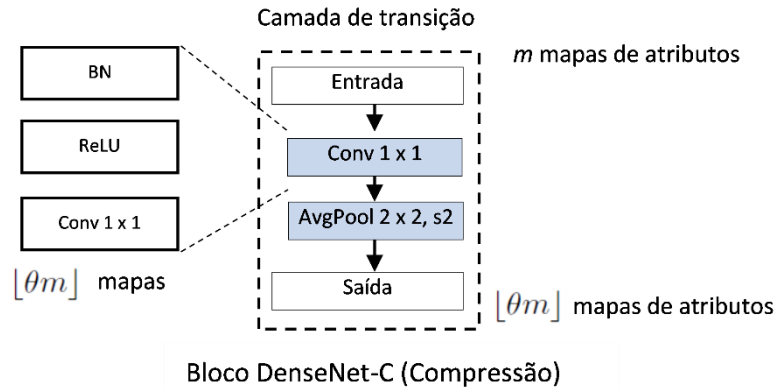


Figura 2.49 – DenseNet com 4 blocos densos e camadas de transição. Adaptado de [HUANG *et al.* 2017].



Para melhorar a compactação nas camadas de transição, reduzindo também o número de canais, o número de  $m$  mapas de atributos de entrada é reduzido por um fator de compressão  $\theta$ , com  $0 < \theta \leq 1$  (Figura 2.50). A rede DenseNet é chamada de DenseNet-BC (*Bottleneck/Compression*), quando as camadas de gargalo e camadas de transição com  $\theta < 1$  são usadas.



**Figura 2.50** – Camada de transição entre dois blocos densos.

As configurações de rede DenseNet-121, 169, 201 e 264 são mostradas na Tabela 2.2. Uma estrutura DenseNet-BC com quatro blocos densos foi usada em experimentos com recortes de  $224 \times 224$  do conjunto ImageNet como entrada. A camada de convolução inicial compreende  $2k$  convoluções de tamanho  $7 \times 7$  com passo 2. O número de mapas de atributos em todas as outras camadas segue a taxa de crescimento  $k$ . No final do último bloco denso, um agrupamento de média global (GAP) é executado e, em seguida, um classificador *softmax* é anexado.

**Tabela 2.2** – Arquiteturas DenseNet para ImageNet, com taxa de crescimento  $k = 32$  e camada “conv” como uma sequência BN-ReLU-Conv [HUANG et al. 2017].

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	$112 \times 112$	$7 \times 7$ conv, stride 2			
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2			
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$ $28 \times 28$	$1 \times 1$ conv $2 \times 2$ average pool, stride 2			
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$ $14 \times 14$	$1 \times 1$ conv $2 \times 2$ average pool, stride 2			
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	$14 \times 14$ $7 \times 7$	$1 \times 1$ conv $2 \times 2$ average pool, stride 2			
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	$1 \times 1$	$7 \times 7$ global average pool 1000D fully-connected, softmax			

### 2.3.7 Comparações entre redes

Como vimos, a maneira mais simples de melhorar o desempenho de redes neurais profundas é aumentar sua capacidade de representação de dados, aumentando a profundidade (número de níveis da rede adicionando camadas de convolução) e largura (número de canais em cada nível, ou seja, mapas de atributos por camada). Com isso, pode-se obter modelos de rede maiores e mais complexos com alto desempenho de classificação.

Entretanto, essa solução apresenta duas desvantagens: (1) aumento do número de parâmetros, o que pode causar sobreajuste de dados se a quantidade de dados rotulados do conjunto de treinamento for insuficiente; (2) aumento do número de operações e, por consequência, do tempo de processamento [SZEGEDY *et al.* 2015]. Porém, a eficiência computacional em tempo real e baixo número de parâmetros ainda são condições importantes para dispositivos móveis com recursos computacionais limitados e aplicações com grandes conjuntos de dados [SZEGEDY *et al.* 2016].

O desenvolvimento de redes neurais convolucionais começou com os modelos de rede LeNet [LECUN 1989; LECUN *et al.* 1989; LECUN *et al.* 1990; LECUN *et al.* 1998], que eram operações de convolução simples para extração de atributos, alternadas com operações de *maxpooling* para subamostragem espacial. Em 2012, essas ideias foram refinadas na arquitetura AlexNet, onde as operações de convolução com funções de ativação ReLU eram repetidas várias vezes entre as operações de *maxpooling*, permitindo que a rede aprendesse atributos mais ricos em cada nível de representação de atributos. De 2014 em diante, com a introdução da arquitetura VGG [SIMONYAN e ZISSERMAN 2015] com 16 e 19 camadas de convolução, houve uma tendência de tornar esse estilo de rede cada vez mais profundo, impulsionado pela competição anual ILSVRC. A partir deste momento, foram desenvolvidas várias arquiteturas de rede da família Inception, cada vez mais profundas, incluindo Inception-v1 (GoogLeNet) [SZEGEDY *et al.* 2015] com 22 camadas, Inception-v2 e v3 [SZEGEDY *et al.* 2016; IOFFE e SZEGEDY 2015] e, por último, Inception-v4 e Inception-ResNet [SZEGEDY *et al.* 2017].

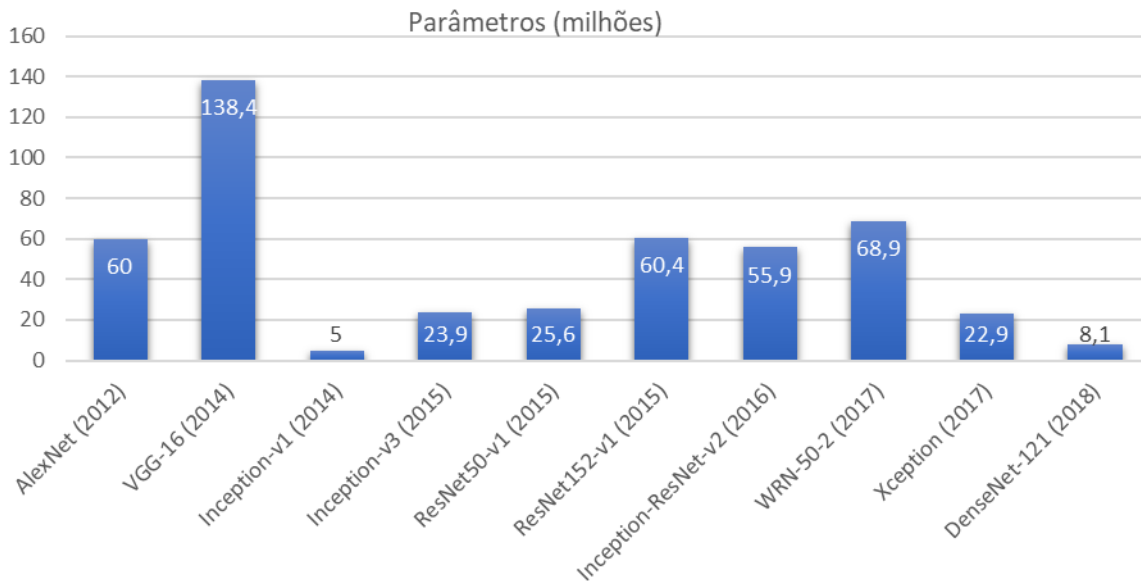
As Figuras 2.51 a 2.53 mostram uma comparação do número de parâmetros, profundidade em camadas e desempenho das redes de classificação no conjunto ImageNet.

A rede VGG-16 tem uma arquitetura simples e bom desempenho, mas apresenta alto custo computacional (138 milhões de parâmetros), ou seja, 2,3 vezes mais parâmetros do que AlexNet, que tem 60 milhões de parâmetros. Por outro lado, Inception-v1 emprega apenas 5 milhões de parâmetros, o que representa uma redução de 12 vezes em relação ao AlexNet. Porém, a complexidade das arquiteturas Inception torna mais difícil fazer alterações na rede para adaptar a arquitetura para novas tarefas [SZEGEDY *et al.* 2016].

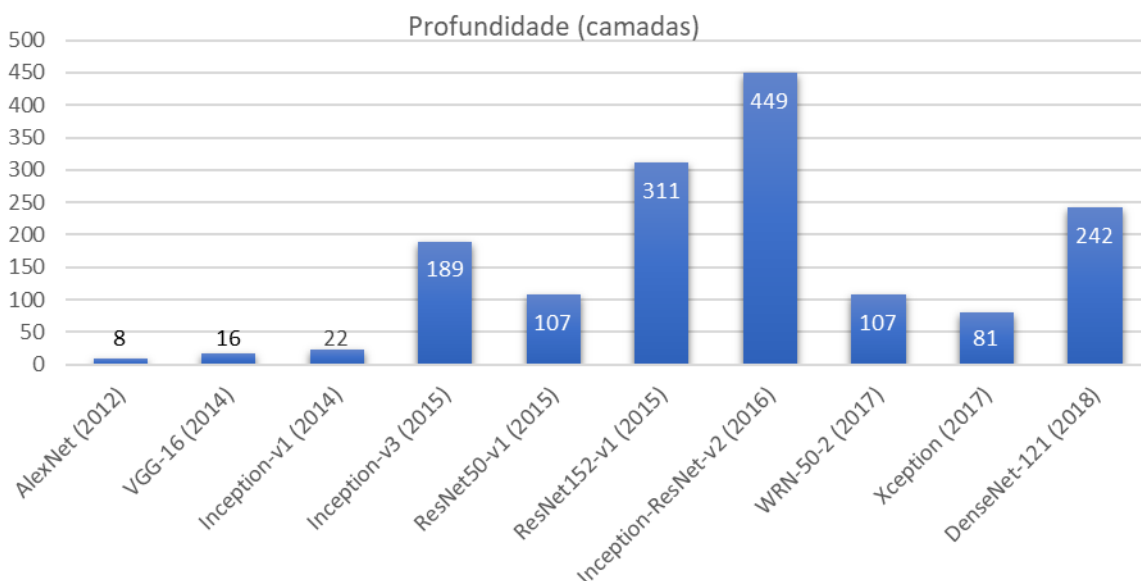
A rede ResNet de 152 camadas, ou seja, 9,5 vezes mais profunda que VGG-16 (mas com menos parâmetros), tem o mesmo número de parâmetros que AlexNet, mas apresenta melhor desempenho no conjunto de dados ImageNet do que AlexNet e VGG-16. Em comparação com as arquiteturas Inception-v3, as redes residuais mostram uma capacidade de generalização semelhante, mas têm mais parâmetros. Uma rede residual larga de 50 camadas de profundidade (WRN-50-2) tem precisão compatível com uma rede residual profunda de 152 camadas (ResNet152-v1), com um número comparável de parâmetros em torno de 66 milhões. Assim, em [ZAGORUYKO e KOMODAKIS 2016] chegou-se à conclusão que o poder de representação das redes residuais está em blocos residuais, e não em profundidade extrema, como afirmado anteriormente.

Comparado ao Inception-v3 com 36 camadas convolucionais, que foi o primeiro colocado no ILSVRC 2015 no conjunto de dados ImageNet, a rede Xception tem aproximadamente o mesmo número de parâmetros e desempenho de classificação no conjunto de dados ImageNet. Além disso, a rede Xception com 36 camadas supera ligeiramente os resultados de redes residuais mais profundas e com mais parâmetros, como ResNet-152 [CHOLLET 2017]. Assim, a melhoria de desempenho não vem do aumento da capacidade, mas de um uso mais eficiente dos parâmetros do modelo.

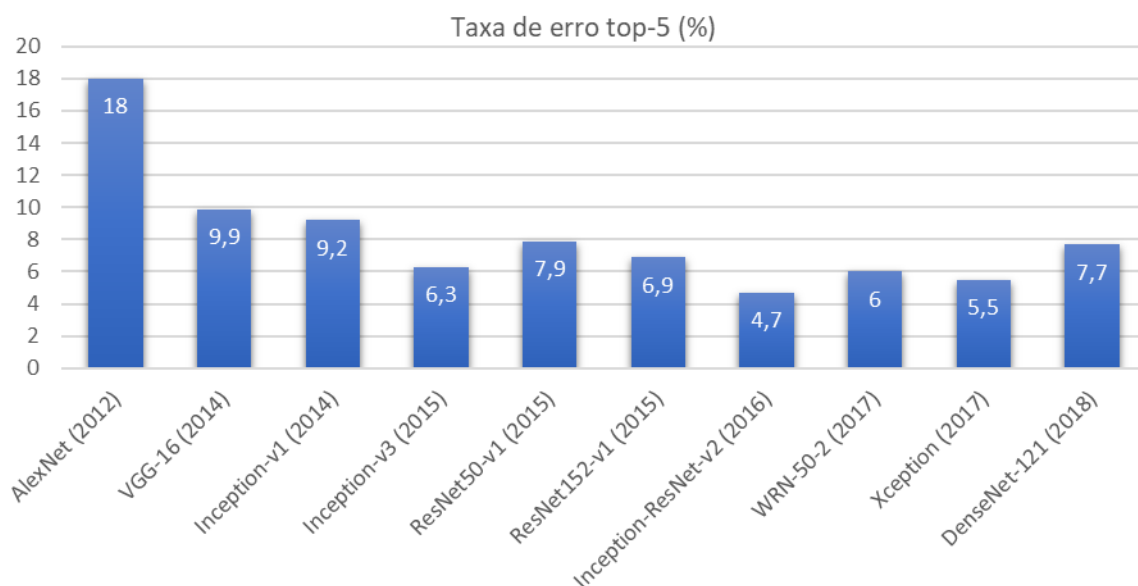
Por fim, a rede DenseNet-121 tem menos parâmetros que Inception-v3, Xception e ResNet-152, mas apresenta um desempenho comparável, embora um pouco inferior.



**Figura 2.51** – Comparação do número de parâmetros das redes de classificação. Fonte: <https://keras.io/api/applications/>



**Figura 2.52** – Comparação da profundidade em número de camadas convolucionais das redes de classificação, incluindo camadas de ativação e BN. Fonte: <https://keras.io/api/applications/>



**Figura 2.53** – Comparação de desempenho de classificação no conjunto ImageNet. Fonte: <https://keras.io/api/applications/>.

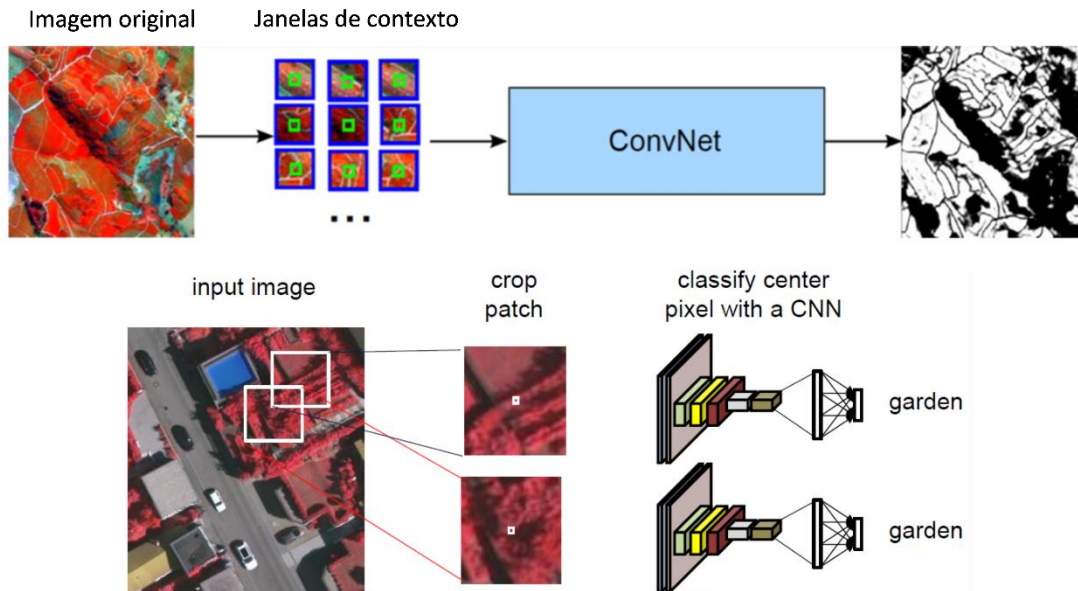
A seguir, descrevemos algumas redes de segmentação semântica baseadas em CNN, que dependem de atributos visuais de alta qualidade aprendidos pelas redes de classificação descritas nesta seção.

## 2.4 Arquiteturas de segmentação semântica

Modelos de segmentação semântica são baseados em adaptações de redes neurais convolucionais projetadas para tarefas de classificação de imagens. Desta forma, os avanços nas CNNs de classificação, tais como VGG [SIMONYAN e ZISSERMAN 2015], ResNet [HE *et al.* 2016a], Xception [CHOLLET 2017] e DenseNet [HUANG *et al.* 2017], impulsionaram melhorias de desempenho não apenas na tarefa de classificação da imagem, mas também na tarefa de segmentação semântica de imagens. No entanto, a segmentação semântica de imagens é um problema de visão computacional, onde cada *pixel* deve ser classificado de acordo com um rótulo de classe do objeto a qual pertence. Sendo assim, ao contrário da classificação, que é um problema de predição esparsa, onde um único rótulo é atribuído a uma imagem inteira, na segmentação semântica é necessária uma predição densa em termos de *pixels*.

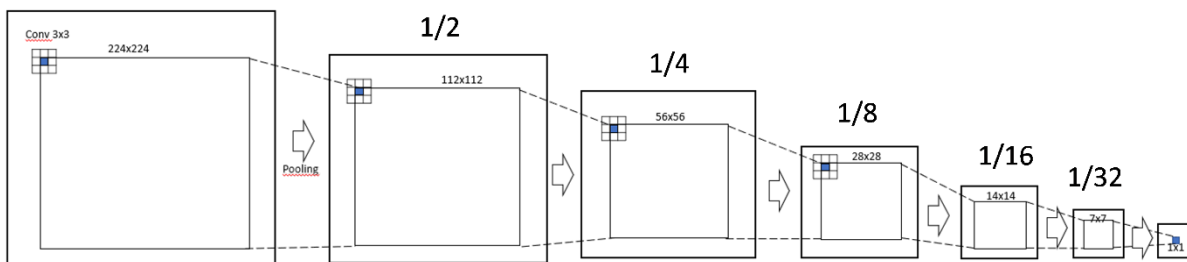
As abordagens iniciais de segmentação semântica usavam redes neurais convolucionais profundas para classificação de *patches*, onde cada *pixel* era rotulado com a classe de uma região ao redor do *pixel*. Desta forma, as redes eram treinadas com uma configuração de janela deslizante que fornece uma região local (*patch*) como entrada, para prever o rótulo de classe de cada *pixel* central [RONNEBERGER *et al.* 2015]. O principal motivo do uso de *patches* na segmentação de imagens foi que as redes de classificação têm camadas totalmente conectadas e, portanto, exigem entrada de tamanho fixo. Além disso, a quantidade de dados de treinamento em termos de *patches* é maior do que o número de imagens de treinamento disponíveis. Uma das principais desvantagens deste método é a falta de eficiência, porque a rede deve ser executada separadamente para cada *patch* e há

muita redundância de contexto devido à sobreposição de *patches*, como mostra a Figura 2.54 [NOGUEIRA *et al.* 2016]. Além disso, há um *trade-off* entre a precisão da localização e o tamanho da janela de contexto, ou seja, *patches* maiores requerem mais camadas de subamostragem que reduzem a precisão da localização, enquanto pequenos *patches* permitem que a rede veja apenas uma pequena área de contexto na imagem [RONNEBERGER *et al.* 2015].



**Figura 2.54** – Segmentação semântica usando CNN para classificação de patches [NOGUEIRA *et al.* 2016].

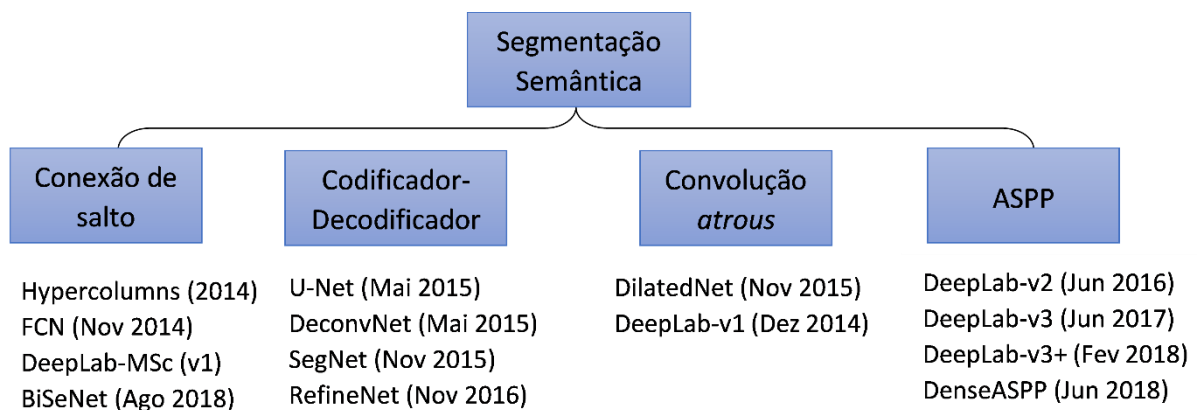
As primeiras abordagens de segmentação semântica densa de *pixels* da imagem inteira ou sub-blocos dela (em vez de classificação de *patches*) foram baseadas em arquiteturas CNNs profundas [FARABET *et al.* 2013], mas apresentaram resultados grosseiros e pouco satisfatórios. Isso ocorre devido às várias camadas de agrupamento/subamostragem (*pooling*) das CNNs, que reduzem significativamente a resolução do mapa de atributos das camadas mais profundas em comparação com as dimensões originais da imagem de entrada. Por exemplo, na rede VGG são usadas cinco camadas de subamostragem (*maxpooling*  $2 \times 2$  não sobreposta com passo 2) e, portanto, o tamanho do mapa de atributos na saída da rede tem um fator de saída de 32 ( $1/32$  da dimensão de entrada), como mostra a Figura 2.55.



**Figura 2.55** – Tamanho dos mapas de atributos nos blocos VGG-16 com campo receptivo de  $224 \times 224$  (com *same padding*) e fator de saída de 32.

Na [Figura 2.55](#), as sucessivas camadas de *pooling* na CNN aumentam o campo de visão na camada final, sendo capaz de agregar o contexto na imagem de entrada com um campo receptivo de  $224 \times 224$  *pixels*, porém descartam as informações de localização com a redução da resolução. No entanto, a segmentação semântica requer alta resolução dos mapas de previsão de classe e, deste modo, precisa que as informações de localização sejam preservadas. Sendo assim, uma técnica de *upsampling* deve ser utilizada para aumentar a amostragem do mapa de atributos da camada mais profunda, para corresponder às dimensões da imagem de entrada. Isto pode ser feito replicando-se atributos dentro de um bloco de saída  $32 \times 32$  ou usando interpolação bilinear, porém, neste caso, todos os *pixels* dentro de um bloco geram resultados grosseiros de previsão da segmentação semântica.

Existem quatro categorias principais de arquiteturas de redes de segmentação semântica que evoluíram na literatura para lidar com estes problemas, como mostra a [Figura 2.56](#).



**Figura 2.56** – Categorias de arquiteturas de segmentação semântica de imagens.

- 1) Arquitetura de salto – para compensar a baixa resolução de atributos de alto nível, mapas de atributos de camadas intermediárias também são usados através de conexões de salto. Algumas redes deste tipo são:
  - Hypercolumns [[HARIHARAN et al. 2014a](#)];
  - FCN (*Fully Convolutional Networks*) [[LONG et al. 2015](#); [SHELHAMER et al. 2017](#)];
  - BiSeNet [[YU et al. 2018](#)];
  - DeepLab-MSc (v1) [[CHEN et al. 2015](#)].
- 2) Arquitetura codificador-decodificador – o codificador reduz gradualmente a dimensão espacial com camadas de *pooling* e o decodificador recupera gradualmente a dimensão espacial. Geralmente, há conexões de atalho (índices de *pooling*) ou salto (mapas de atributos) entre o codificador e o decodificador para ajudar o decodificador a recuperar os detalhes do objeto.
  - U-Net [[RONNEBERGER et al. 2015](#)];
  - DeconvNet [[NOH et al. 2015](#)];
  - SegNet [[BADRINARAYANAN et al. 2015](#); [BADRINARAYANAN et al. 2017](#)];
  - RefineNet [[LIN, MILAN et al. 2017](#)];
  - FC-DenseNet [[JEGOU et al. 2017](#)].



- 3) Arquitetura com convoluções dilatadas (*atrous*) em série – algumas camadas de *pooling* são eliminadas para preservar a resolução e, em seguida, as convoluções *atrous* permitem extrair mapas de atributos mais densos com maior campo receptivo.
  - DilatedNet [YU e KOLTUN 2016];
  - DeepLab-v1 [CHEN *et al.* 2015].
  
- 4) Arquitetura com convoluções *atrous em paralelo* (ASPP) – usam convoluções *atrous* paralelas com diferentes taxas de dilatação para extrair atributos multiescala.
  - DeepLab-v2, v3 e v3+ [CHEN *et al.* 2018a, CHEN *et al.* 2017, CHEN *et al.* 2018b];
  - DenseASPP [YANG *et al.* 2018].

Além destas redes descritas nas próximas seções, podemos citar outras, tais como: Attention to Scale [CHEN *et al.* 2016a], LRR [GHIASI e FOWLKES 2016] com pirâmide laplaciana [BURT e ADELSON 1983], ENet [PASZKE *et al.* 2016], FRRN [POHLEN *et al.* 2017], ResNet-38 [WU *et al.* 2019], PSPNet [ZHAO *et al.* 2017], LargeKernel [PENG *et al.* 2017], ICNet [ZHAO *et al.* 2018], TU-SimpleDUC [WANG *et al.* 2018], ExFuse [ZHANG *et al.* 2018b], PAN [LI *et al.* 2018], MRFN [YUAN *et al.* 2020], CASINet [JIN *et al.* 2020], etc.

Todas estas redes são treinadas em um conjunto de dados de referência para tarefas de segmentação semântica, como *PASCAL Visual Object Classes* (PASCAL-VOC) 2012 [EVERINGHAM *et al.* 2015] e *Microsoft Common Objects in Context* (MS-COCO) [LIN, MAIRE *et al.* 2014].

O conjunto PASCAL-VOC contém a classe de segundo plano (fundo) e 20 categorias de objetos de primeiro plano, incluindo: avião, bicicleta, barco, ônibus, carro, motocicleta, trem, garrafa, cadeira, mesa de jantar, vaso de planta, sofá, TV/monitor, pássaro, gato, vaca, cachorro, cavalo, ovelha e pessoa. No entanto, a maior parte desse conjunto de imagens tem uma ou duas classes de primeiro plano cercadas por um plano de fundo altamente variado, como mostra a Figura 2.57. Cada imagem tem anotações de segmentação em nível de *pixel*, anotações de caixa delimitadora e anotações de classe de objeto. Este conjunto é dividido em três subconjuntos: 1.464 imagens para treinamento, 1.449 imagens para validação e um conjunto de teste privado.

Outros conjuntos de dados estão disponíveis, como PASCAL-Context [MOTTAGHI *et al.* 2014]; PASCAL Person-Part [CHEN *et al.* 2014]; SUN RGB-D [SONG *et al.* 2015]; ADE20K [ZHOU *et al.* 2014] e NYU [SILBERMAN *et al.* 2012] – para análise semântica de cenas internas em aplicações de realidade aumentada; e Cityscapes [CORDTS *et al.* 2016], CamVid [BROSTOW *et al.* 2009] e KITTI [GEIGER *et al.* 2012] – para cenas externas em aplicações de direção autônoma.

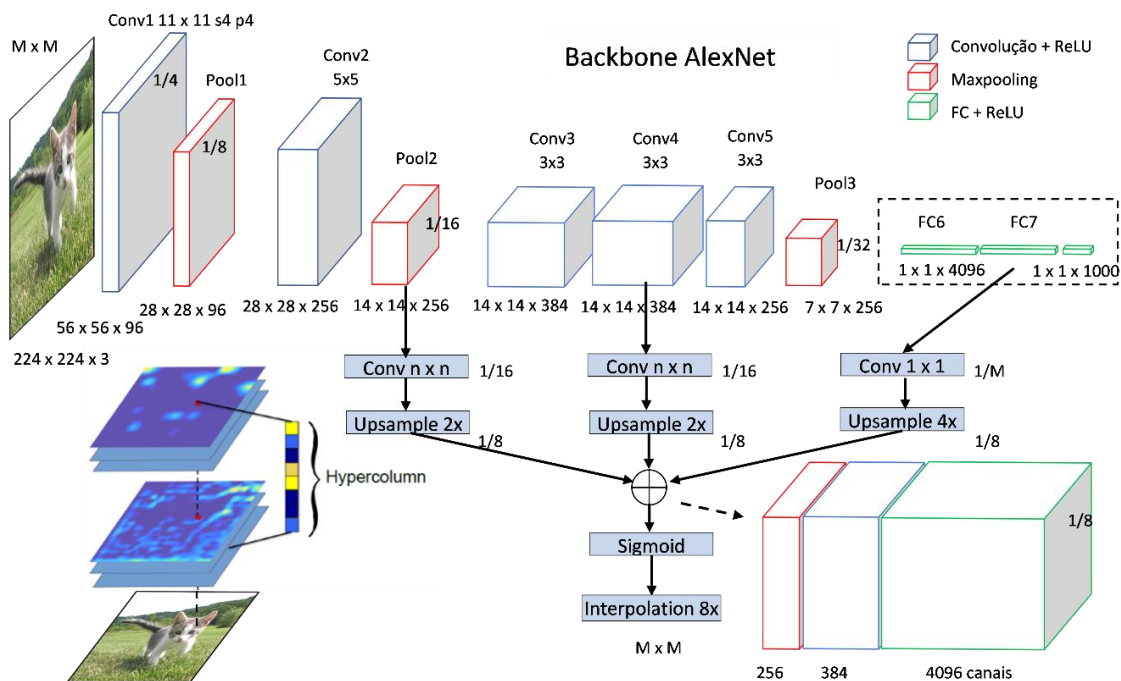


Figura 2.57 – Exemplos de imagens do conjunto PASCAL-VOC [EVERINGHAM *et al.* 2015].

## 2.4.1 Modelo Hypercolumns

Algoritmos de classificação baseados em CNNs usam a saída da última camada como uma representação de atributos de alto nível para previsão da classe dos objetos. No entanto, as informações nesta camada são espacialmente grosseiras, sem informações de localização precisas. Já as camadas inferiores com campos receptivos menores têm informações precisas de localização, mas não capturam a semântica. Para atender a estes dois requisitos, Hariharan *et al.* desenvolveram uma rede Hypercolumns em 2014 [HARIHARAN *et al.* 2014a], que gera uma hipercoluna em um *pixel* da entrada, ou seja, um vetor de ativações de todas as unidades dos mapas de atributos de diferentes camadas da CNN que se localizam acima desse *pixel*, como mostra a Figura 2.58.

A rede Hypercolumns usa conexões de salto para fundir mapas de atributos de maior resolução das camadas intermediárias de uma CNN-base (*backbone*), com os mapas de atributos de baixa resolução da saída. Camadas convolucionais  $n \times n$  são adicionadas com qualquer tamanho de *kernel* ( $11 \times 11$ ,  $5 \times 5$  e  $3 \times 3$ ), embora na camada totalmente conectada (FC), que produz mapas de atributos  $1 \times 1$ , a convolução adicional é restrita a  $1 \times 1$ . Devido às operações de subamostragem na CNN, os mapas de atributos das camadas intermediárias e da camada de saída não têm a mesma resolução que a entrada. Para obter uma localização precisa, os mapas de atributos das camadas superiores são sobreamostrados com interpolação bilinear de 2x ou 4x. Em seguida, os mapas de atributos são concatenados, gerando um vetor de hipercoluna para cada *pixel* local. Por exemplo, na Figura 2.58, os ramos sobreamostrados e concatenados de Pool2 (256 canais), Conv4 (384 canais) e FC7 (4.096 canais) da rede-base AlexNet [KRIZHEVSKY *et al.* 2012] geram um mapa de  $28 \times 28$  com 4.736 canais. Por fim, esse mapa de saída passa por uma função de ativação não linear sigmóide, que é novamente sobreamostrado para obter o mapa de previsão final. Como o fator de redução da resolução espacial na saída é 8, uma interpolação de 8x é usada para recuperar a resolução original. Em vez de treinar a rede do zero, uma rede pré-treinada é ajustada.



**Figura 2.58** – Arquitetura de rede Hypercolumns com os mapas de atributos do backbone AlexNet e camadas convolucionais e de sobreamostragem adicionais (caixas azuis).



### 2.4.2 Modelo FCN

A rede FCN (*Fully Convolutional Network*) [LONG *et al.* 2015] popularizou o uso de arquiteturas CNNs profundas, normalmente usadas para classificação de imagens, adaptando-as para realizar previsões densas a nível de *pixel* na tarefa de segmentação semântica de imagens. Para isso, as CNNs primeiro são convertidas em redes totalmente convolucionais, ou seja, todas as camadas totalmente conectadas (FC) são transformadas em camadas convolucionais. Essas versões de redes totalmente convolucionais recebem entradas de tamanho arbitrário e preveem saídas densas de tamanho correspondente, embora subamostradas, produzindo mapas de saída grosseiros. Em contraste com a abordagem de classificação de *patches*, tanto o aprendizado quanto a inferência são executados na imagem inteira, por computação *feedforward* e retropropagação densa.

Na época, três CNNs foram testadas como *backbone* das redes FCN. Foram consideradas a arquitetura AlexNet [KRIZHEVSKY *et al.* 2012], que ganhou o ILSVRC-2012, bem como VGG [SIMONYAN e ZISSERMAN 2015] e GoogLeNet [SZEGEDY *et al.* 2015]) que tiveram bom desempenho no ILSVRC-2014. Estas redes de classificação de imagens foram adaptadas, reinterpretando-se as camadas FC como camadas convolucionais. A seguir é explicado como as CNNs, em particular a rede VGG-16, são convertidas em redes totalmente convolucionais.

#### Convolucionalização da CNN

As CNNs recebem entradas de tamanho fixo e produzem saídas não espaciais, uma vez que as camadas classificadoras totalmente conectadas (FC) dessas redes têm dimensões fixas e descartam as coordenadas espaciais, como mostra a Figura 2.59. No entanto, essas camadas totalmente conectadas também podem ser vistas como convoluções com *kernels* que cobrem toda a região de entrada, como ilustrado na Figura 2.60.

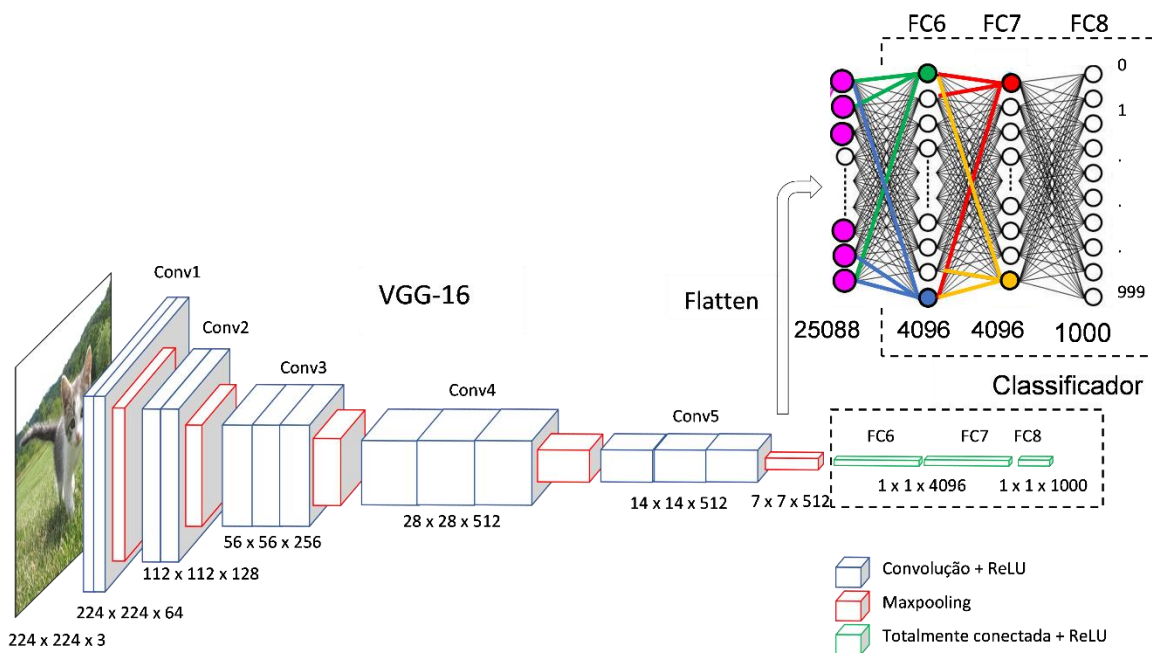
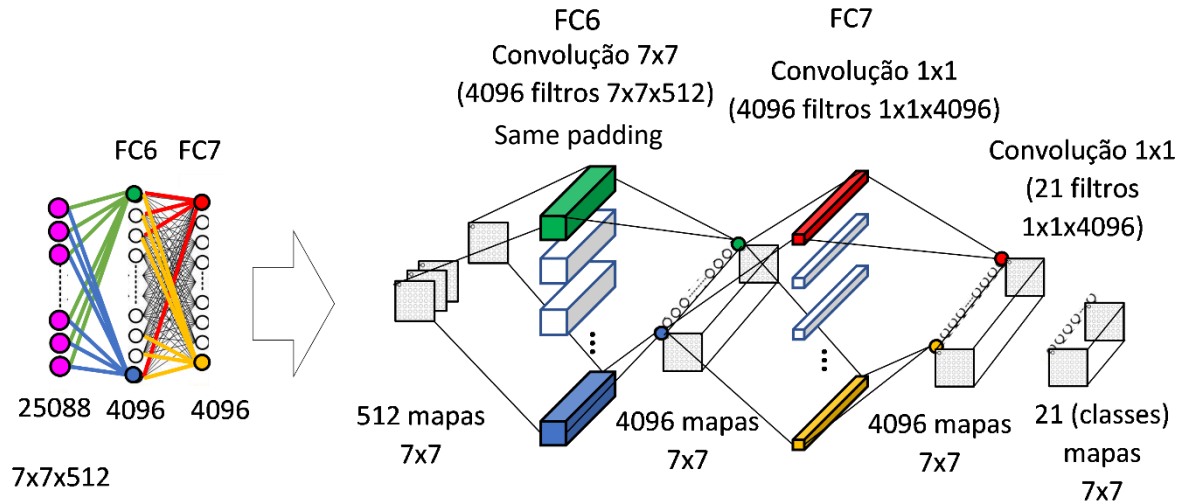


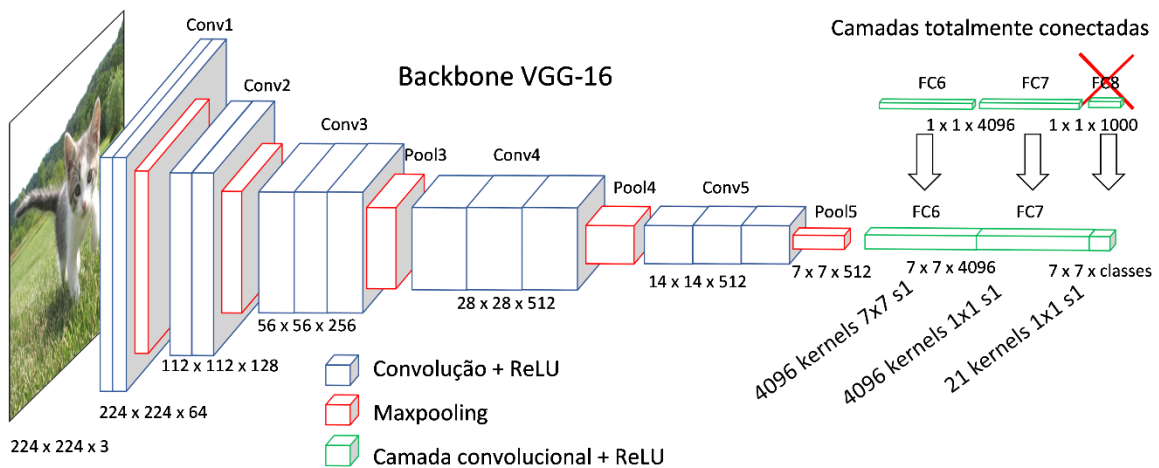
Figura 2.59 – Mapas de atributos de VGG-16 com camadas totalmente conectadas.



**Figura 2.60** – Convolutionalização das camadas totalmente conectadas.

Para convolutionalização das camadas totalmente conectadas de VGG, primeiro é descartada a camada classificadora final FC8, com 1.000 rótulos de classe de ImageNet. Em seguida, todas as camadas totalmente conectadas são convertidas em camadas convolucionais. A primeira camada FC6 é substituída por uma camada de convolução  $7 \times 7$  com preenchimento (*same padding*). A camada FC7 é substituída por uma camada convolucional  $1 \times 1$ . Na última camada, uma convolução  $1 \times 1$  com número de *kernels* igual ao número de classes é anexada para prever pontuações (*scores*) para cada uma das 21 classes de PASCAL-VOC [EVERINGHAM *et al.* 2015], incluindo a classe de fundo, em cada uma das unidades de saída grosseira dos 21 mapas de atributos. A Figura 2.60 mostra a convolutionalização da rede da Figura 2.61, quando a entrada tem tamanho  $224 \times 224$ , gerando uma saída com  $C = 21$  mapas subamostrados com tamanho correspondente  $7 \times 7$ .

Portanto, todas as três camadas FC são substituídas por camadas de convolução, com duas vantagens principais: primeiro, os atributos podem ser extraídos sem descartar totalmente as informações de localização de cada *pixel*; e, segundo, permite entradas de tamanhos arbitrários que geram mapas de saída de classificação com tamanho subamostrado correspondente, em contraste com as redes CNN originais, que exigem entradas de tamanho fixo e geram saídas não espacial [LONG *et al.* 2015].

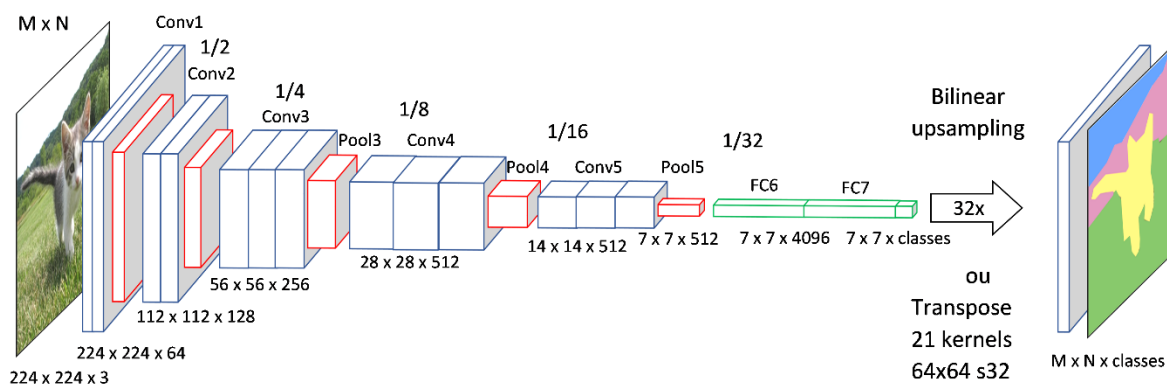


**Figura 2.61** – CNN-base (VGG-16) convertida em rede totalmente convolucional.

## Redes FCN-32s, 16s e 8s

Na [Figura 2.62](#), a entrada da rede FCN é uma imagem com tamanho espacial de  $M \times N$  *pixels* e dimensão de canal com três canais de cores RGB. Cada camada em uma CNN gera uma matriz tridimensional de tamanho  $h \times w \times d$ , onde  $h$  e  $w$  são dimensões espaciais e  $d$  é a dimensão de canal do mapa de atributos. Uma FCN opera em uma entrada de qualquer tamanho e produz um mapa de saída de dimensões espaciais correspondentes, mas as dimensões de saída são normalmente reduzidas por operações de subamostragem (*pooling*) dentro da rede. Isso torna a saída de uma FCN mais grosseira, reduzindo a resolução original da entrada por um fator igual ao passo de saída dos campos receptivos das unidades de saída. Assim, se a imagem de entrada for de tamanho  $M \times N$ , após a primeira camada de subamostragem com passo 2, o tamanho do mapa de atributos torna-se metade do tamanho de entrada original, ou seja,  $M/2 \times N/2$ , e assim por diante. Depois de passar por cinco subamostragens de passo 2 até FC7, o tamanho da saída é  $M/32 \times N/32$ , com fator total de saída da rede igual a 32 [[LONG et al. 2015](#)].

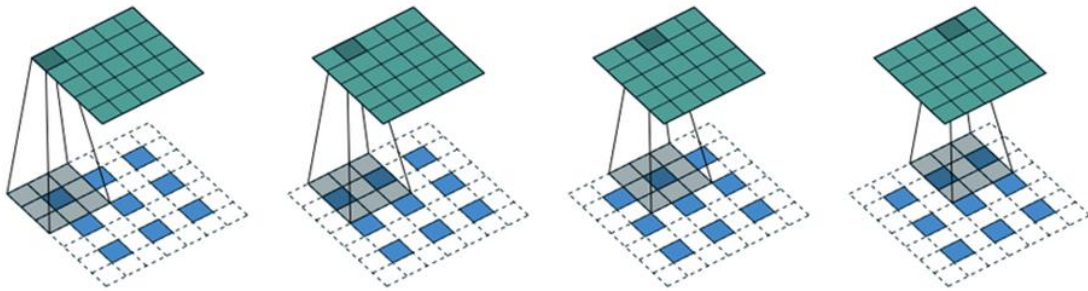
Desta forma, depois de convolucionar camadas totalmente conectadas, os mapas de atributos da saída ainda precisam ser submetidos a uma operação de deconvolução para sobreamostragem de volta à resolução da imagem original. Na [Figura 2.62](#), como a FCN tem um fator de saída 32 (*stride* 32), uma camada de deconvolução com *upsampling* de 32x é necessária para recuperar o mesmo tamanho da imagem de entrada. As camadas de *upsampling* possibilitam a aprendizagem de ponta a ponta usando uma CNN-base com operações de subamostragem (*pooling*), permitindo uma predição a nível de *pixel* na mesma resolução da entrada.



**Figura 2.62** – Mapas de atributos da rede FCN-32s com rede-base VGG-16.

Uma maneira de sobreamostrar saídas grosseiras para *pixels* densos é usando a interpolação bilinear. Desta forma, uma camada de deconvolução é adicionada para fazer uma sobreamostragem ou *upsampling* bilinear das saídas grosseiras para saídas com maior densidade de *pixels*.

Alternativamente, em vez de usar uma deconvolução fixa com interpolação bilinear, as camadas deconvolucionais aprendem a fazer *upsampling* usando uma convolução transposta, como mostra a [Figura 2.63](#). Assim, a convolução transposta [[DUMOULIN e VISIN 2016](#)] permite fazer a interpolação dos mapas de atributos com diferentes *strides* para sobreamostragem de  $s$  vezes (com  $s > 1$ ), onde os parâmetros da deconvolução podem ser aprendidos (pesos do filtro). A convolução transposta na [Figura 2.63](#) é implementada como uma convolução normal  $3 \times 3$  com passo 1 ([Apêndice D.2](#)).

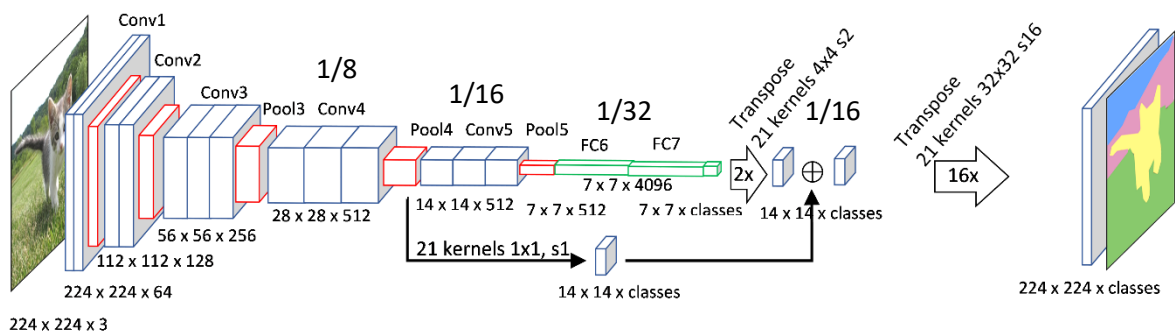


**Figura 2.63** – *Convolução transposta para upsampling de 2x da entrada  $3 \times 3$  (azul) para uma saída  $5 \times 5$  (verde), com kernel  $3 \times 3$  e passo  $s = 2$  ( $s - 1 > 0$ : número de zeros inseridos entre elementos da entrada para obter sobreamostragem de  $s$  vezes). Implementação como convolução normal com kernel  $3 \times 3$  com passo  $s = 1$  [DUMOULIN e VISIN 2016].*

Na [Figura 2.62](#), a rede FCN-32s de fluxo único faz uma previsão com uma sobreamostragem de 32x em uma única etapa, usando uma convolução transposta com  $C = 21$  kernels de  $64 \times 64$  com passo  $s = 32$  (ou interpolação bilinear fixa). Infelizmente, o *upsampling* com interpolação bilinear de 32x, ou mesmo com camadas deconvolucionais usando convolução transposta, produz mapas de segmentação grosseiros devido à perda de informações durante as subamostragens. Portanto, a saída desta rede, com passo de 32 *pixels* na camada de predição final, limita a precisão de detalhes na saída ampliada.

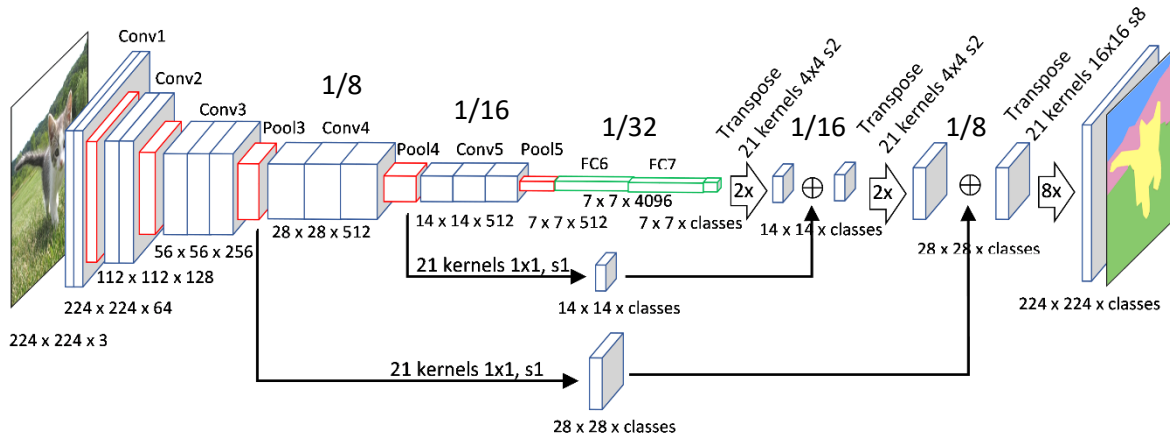
Por este motivo, uma nova arquitetura que combina informações semânticas grosseiras de alto nível (da camada profunda da CNN) com informações refinadas de baixo nível (de camadas intermediárias e superficiais) foi construída para refinar a previsão, produzindo segmentações detalhadas. Assim, a camada de previsão final é fundida por adição com camadas inferiores com passos mais finos. A vantagem é que há uma fusão de informação de alto nível semântico (no topo da CNN) com informação de baixo nível semântico, porém, de maior resolução espacial.

Na [Figura 2.64](#), a combinação de previsões da camada final da CNN ( $7 \times 7 \times C$  classes) sobreamostrada em 2x e da camada Pool4 ( $14 \times 14 \times C$  classes), com posterior sobreamostragem de 16x, permite que a rede FCN-16s retenha informações semânticas de alto nível e preveja detalhes mais finos. Primeiro, o passo de saída é dividido pela metade (32 para 16) adicionando uma camada de *upsampling* de 2x usando convolução transposta, com  $C$  filtros  $4 \times 4$  e  $s = 2$ , onde  $C$  é o número de classes da tarefa de segmentação para obter mapas de  $14 \times 14 \times C$ . Depois, uma camada de convolução  $1 \times 1$  é adicionada no topo do Pool4 para produzir previsões de classe adicionais ( $14 \times 14 \times C$ ). Essas saídas são fundidas somando-se os dois mapas de previsões. Por fim, as previsões combinadas de  $14 \times 14 \times C$  são submetidas a uma convolução transposta de 16x ( $C$  kernels  $32 \times 32$  e  $s = 16$ ), de volta ao tamanho da imagem original de entrada ( $224 \times 224 \times C$ ).



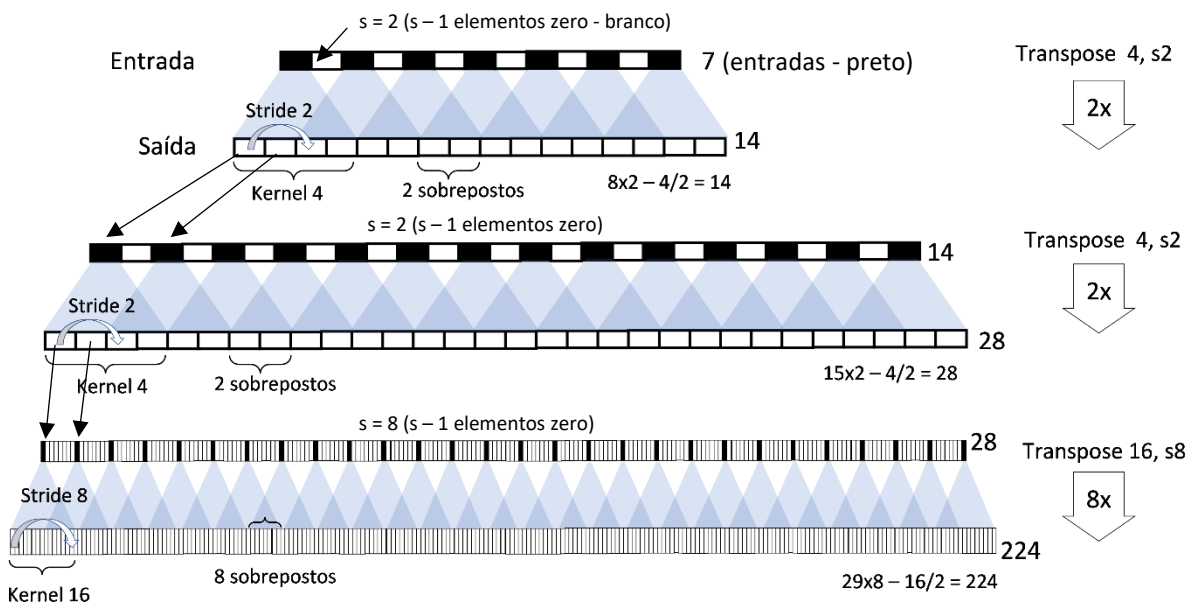
**Figura 2.64** – *Mapas de atributos da rede FCN-16s com rede-base VGG-16.*

Continuando desta forma, na rede FCN-8s mostrada na [Figura 2.65](#), as previsões adicionais a partir do Pool3, com passo de saída 8 ( $28 \times 28 \times C$  classes), são fundidas com um *upsampling* de 2x das previsões fundidas do Pool4 e FC7, para obter mapas de  $28 \times 28 \times C$ . Finalmente, as previsões de  $28 \times 28 \times C$  são sobreamostradas com convolução transposta de 8x ( $C$  kernels de  $16 \times 16$  e  $s = 8$ ), de volta ao tamanho original da imagem ( $224 \times 224 \times C$ ). Portanto, essa arquitetura de segmentação FCN usa três camadas de deconvolução com convolução transposta para aprender a aumentar a amostragem e obter uma previsão refinada.



**Figura 2.65** – Mapas de atributos da rede FCN-8s com rede-base VGG-16.

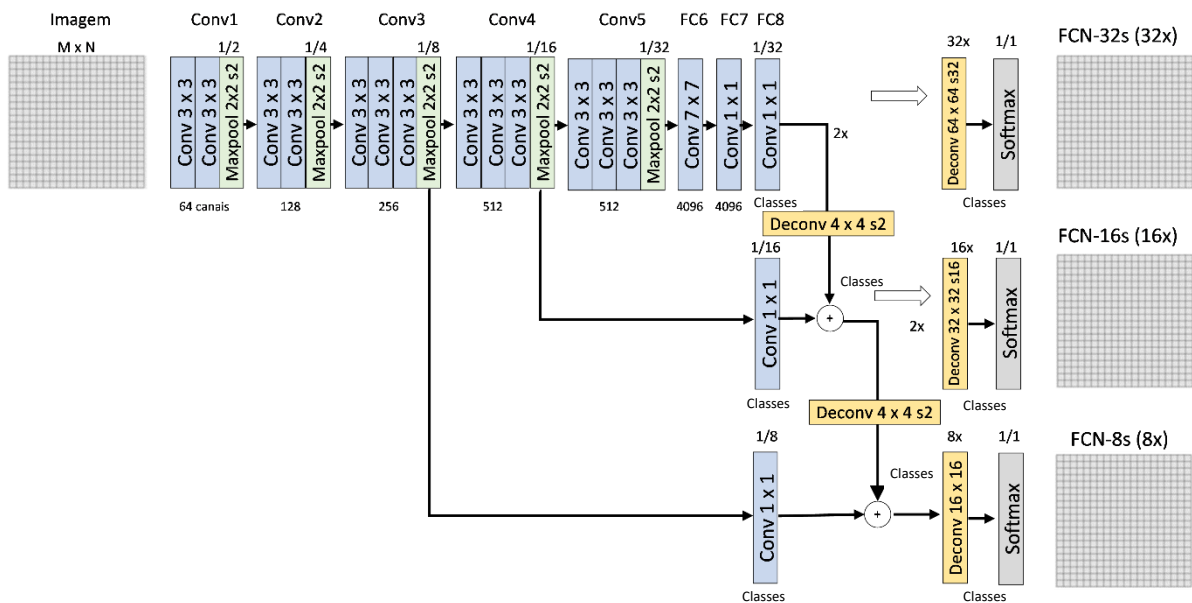
A [Figura 2.66](#) mostra os efeitos desta sequência de três convoluções transpostas 1D consecutivas com fatores 2x, 2x e 8x (correspondente a um fator total de saída de 32), sobre um vetor de atributos de tamanho 7, para recuperar a resolução espacial original da entrada de 224 pixels.



**Figura 2.66** – Sequência de deconvoluções com três convoluções transpostas.

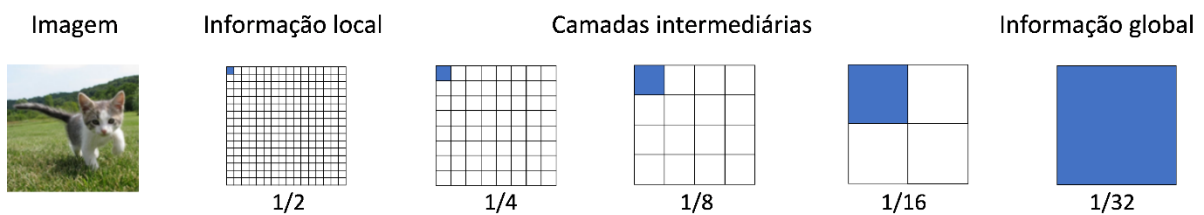


O refinamento de redes totalmente convolucionais, fundindo informações de múltiplas camadas intermediárias dentro da rede, usando conexões de salto, melhora os detalhes de segmentação. A [Figura 2.67](#) mostra a estrutura básica das redes FCNs.



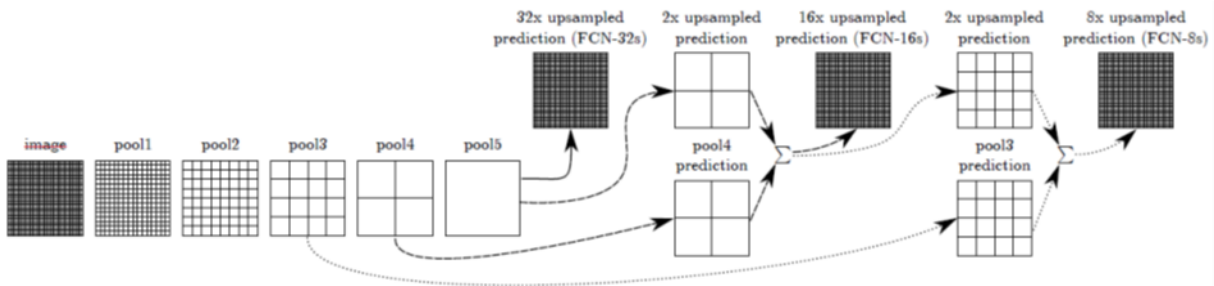
**Figura 2.67** – Arquitetura das redes FCN-32s, 16s e 8s com deconvolução transposta.

Como vimos, a segmentação semântica enfrenta uma tensão entre semântica e localização. A informação de alto nível – de baixa resolução, mais abstrata e global – das camadas mais profundas resolve a semântica, enquanto a informação de baixo nível – de maior resolução, menos abstrata e local – das camadas mais rasas resolve a localização. Hierarquias de mapas de atributos das CNNs codificam conjuntamente a localização e a semântica em uma pirâmide local para global. Na [Figura 2.68](#), os mapas em cada camada da rede são mostrados como grades que revelam a resolução espacial relativa.



**Figura 2.68** – Hierarquia de atributos com informação local e global.

As redes FCNs introduziram conexões de salto (*skip connections*) para aumentar a granularidade das camadas mais profundas. Assim, a rede combina camadas de múltiplas resoluções da hierarquia de atributos, usando as conexões de salto para refinar a precisão espacial da saída e melhorar o desempenho da predição espacialmente densa. As redes FCN aprendem a combinar atributos refinados de camadas rasas com atributos grosseiros da camada profunda, que possui informações semânticas abstratas mais complexas. Portanto, uma arquitetura de salto permite combinar informações semânticas (global) e informações refinadas (local), como mostra a [Figura 2.69](#). Somente camadas de *pooling* e predição sobreamostrada são mostradas. As camadas de convolução intermediárias (incluindo as camadas totalmente conectadas convertidas) são omitidas [LONG *et al.* 2015].

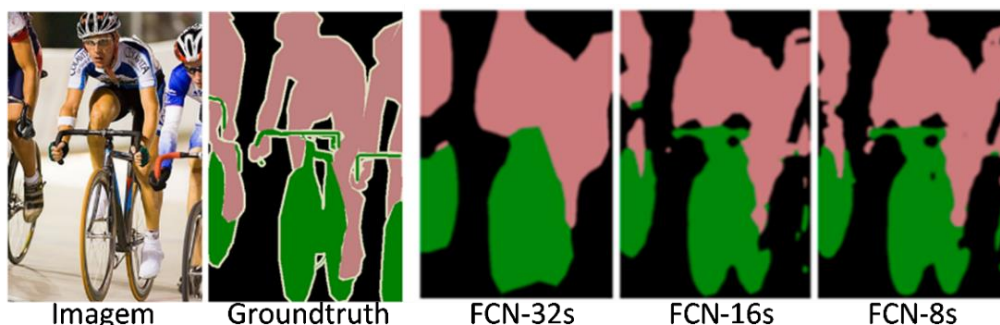


**Figura 2.69** – Combinação de informação de atributos local e global [LONG *et al.* 2015].

Além da rede VGG-16, outras redes de classificação podem ser estendidas para segmentação semântica. Porém, o treinamento da rede FCN do zero não é viável, considerando o tempo necessário para aprender as redes de classificação. Assim, as redes CNN pré-treinadas no ImageNet [DENG *et al.* 2009], como AlexNet [KRIZHEVSKY *et al.* 2012], VGG [SIMONYAN e ZISSERMAN 2015] e GoogLeNet [SZEGEDY *et al.* 2015] foram adaptadas e ajustadas com *fine tuning*, onde as representações aprendidas pelas CNNs na tarefa de classificação são transferidas à tarefa de segmentação semântica. Comparando os resultados, a rede FCN com VGG-16 obteve o melhor desempenho em [LONG *et al.* 2015].

A rede totalmente convolucional, FCN, atingiu segmentação de ponta no conjunto PASCAL-VOC 2012. Na Figura 2.70, as últimas três imagens mostram um exemplo da saída das redes FCNs de passo (*stride*) 32, 16 e 8. O resultado do FCN-32s é muito impreciso devido à perda de informações de localização, enquanto o FCN-8s tem o melhor resultado. O refinamento de redes totalmente convolucionais, ao fundir informações de camadas com diferentes passos de saída, melhora os detalhes de segmentação [LONG *et al.* 2015]. No entanto, o FCN tende a falhar na previsão de objetos muito grandes ou pequenos devido ao seu campo receptivo de tamanho fixo.

As redes de segmentação semântica FCN foram as primeiras a serem treinadas de ponta a ponta para previsão a nível de *pixel* com pré-treinamento supervisionado. Trabalhos anteriores, como SDS [HARIHARAN *et al.* 2014], R-CNN [GIRSHICK *et al.* 2014], [FARABET *et al.* 2013] e [PINHEIRO e COLLOBERT 2014], aplicavam CNNs usando outras abordagens híbridas. Ao contrário destas redes, as arquiteturas CNNs profundas de classificação de imagem foram adaptadas e estendidas na rede FCN, usando modelos-base pré-treinados e transferência de aprendizado para tarefas de segmentação. A FCN foi ajustada com *fine-tuning* para aprender de forma simples e eficiente a partir de imagens inteiras de entrada. Na época de seu desenvolvimento, a rede FCN superou as técnicas do estado da arte em segmentação semântica [LONG *et al.* 2015].



**Figura 2.70** – Resultado de segmentação semântica das redes FCN-32s, FCN-16s e FCN-8s de uma imagem do conjunto PASCAL-VOC [LONG *et al.* 2015].

### 2.4.3 Modelo U-Net

As primeiras abordagens de segmentação semântica adotaram arquiteturas projetadas para previsão de classe em problemas de rotulagem a nível de *pixel*, convertendo uma CNN em uma rede totalmente convolucional (FCN) [LONG *et al.* 2015; SHELHAMER *et al.* 2017]. Os resultados, embora muito encorajadores, foram grosseiros, principalmente porque a subamostragem da CNN reduz a resolução do mapa de atributos.

A partir de 2015 foi desenvolvida um modelo de rede codificador-decodificador para ampliar os mapas de atributos de volta à resolução original de entrada para previsão a nível de *pixel*. Na Figura 2.71, a arquitetura da rede U-Net [RONNEBERGER *et al.* 2015], usada em aplicações biomédicas, consiste em um codificador – caminho de contração (lado esquerdo) para capturar o contexto – e um decodificador – caminho de expansão (lado direito) – que permite uma localização precisa. O caminho expansivo é mais ou menos simétrico ao caminho de contração e produz uma arquitetura em forma de U. Esta rede pode ser treinada de ponta a ponta a partir de poucas imagens de treinamento para segmentação semântica.

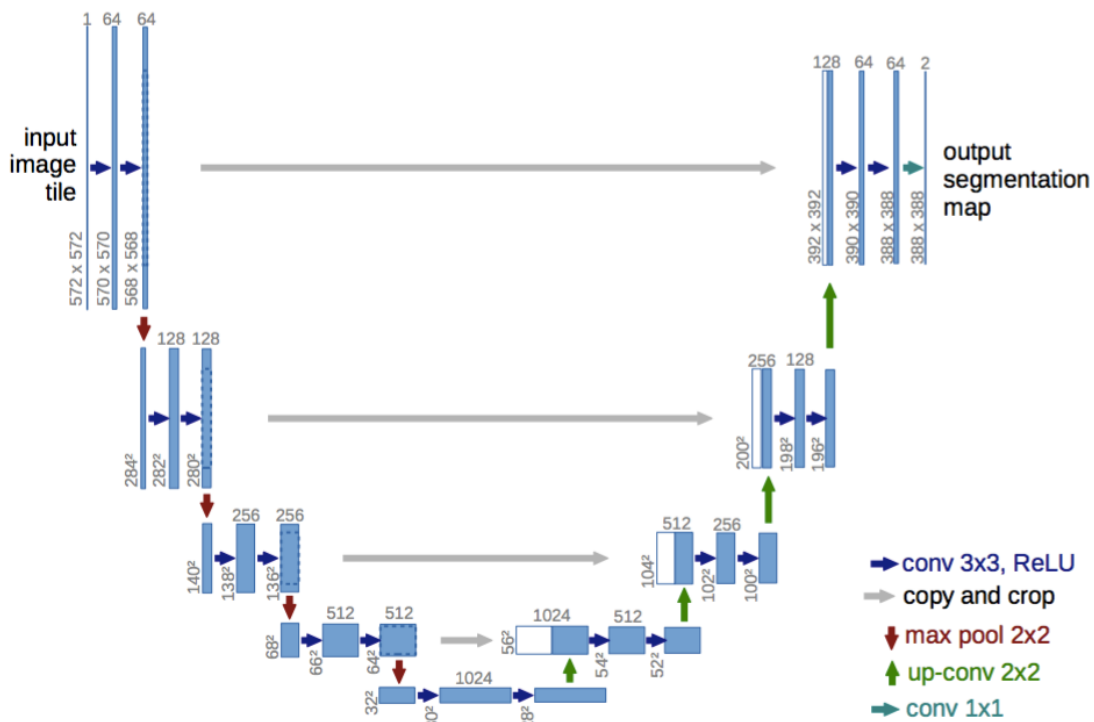


Figura 2.71 – Mapas de atributos da rede U-Net [RONNEBERGER *et al.* 2015].

Cada caixa azul na Figura 2.71 corresponde a um mapa de atributos multicanal passando por uma série de transformações. A altura da haste mostra um tamanho espacial relativo do mapa em  $m \times n$  pixels, enquanto a largura é proporcional ao número de canais. O tamanho  $m \times m = m^2$  do mapa de atributos é indicado na borda esquerda inferior da caixa; o número de canais, na parte superior. As caixas brancas representam mapas de atributos concatenados através de uma conexão de salto. As setas denotam as diferentes operações: convolução  $3 \times 3$  (azul);  $\max$  pooling  $2 \times 2$  com passo 2 (vermelho); e deconvolução ( $\text{upsampling}$  + convolução  $2 \times 2$ ) (verde). As setas cinzas (conexões de salto longo) mostram a transferência de informações de cada camada de codificação para concatenação na camada de decodificação correspondente [RONNEBERGER *et al.* 2015].

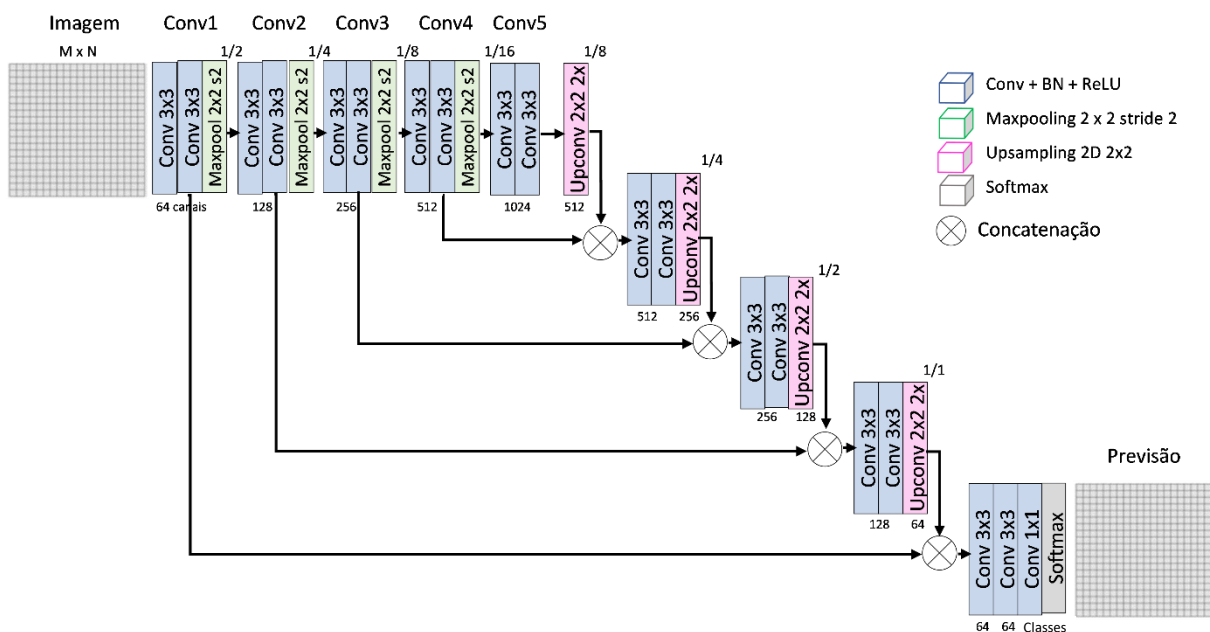


O caminho de contração segue a arquitetura de uma CNN. Na [Figura 2.26](#), a rede neural VGG-13 sem camadas totalmente conectadas é usada como codificador da rede U-Net original, mas outras CNNs podem ser usadas. Este caminho consiste na aplicação repetida de duas convoluções  $3 \times 3$  (convoluções sem preenchimento – *no padding*), cada uma seguida por uma ReLU (setas azuis) e uma operação de *maxpooling*  $2 \times 2$  com passo 2 para subamostragem (setas vermelhas). Em cada etapa de redução da resolução espacial, o número de canais dos mapas de atributos dobra. Devido às convoluções sem preenchimento, a imagem de saída é menor do que a entrada por uma largura de borda constante. Por isso, a cada convolução  $3 \times 3$ , o tamanho do mapa perde dois *pixels* na altura e largura. Desta forma, a rede usa apenas a parte válida de cada convolução, ou seja, o mapa de segmentação semântica contém apenas os *pixels* para os quais o contexto completo está disponível na imagem de entrada [[RONNEBERGER et al. 2015](#)].

Na etapa de expansão, os operadores de *maxpooling* são substituídos por operadores de deconvolução (setas verdes). Assim, cada etapa no caminho expansivo consiste em uma camada de *up-convolution*  $2 \times 2$ , ou seja, um *upsampling* que aumenta a resolução dos mapas de atributos, seguido de uma convolução  $2 \times 2$  com passo 1, que divide pela metade o número de canais.

Nas conexões de salto, há uma concatenação da saída ampliada de baixa resolução do caminho de expansão com o mapa de atributos recortado de alta resolução (caixa branca), correspondente ao caminho de contração, para melhorar a localização de atributos. O recorte (retângulo tracejado dentro da caixa azul do caminho de contração) é necessário devido à perda de *pixels* de borda em cada convolução. Além disso, duas convoluções  $3 \times 3$  sucessivas (seta azul) no caminho de expansão, cada uma seguida por ReLU, aprendem a prever uma saída mais precisa com base nessas informações.

Na camada final, uma convolução  $1 \times 1$  é usada para mapear cada vetor de atributos de 64 canais para um número desejado de classes. No total, a rede U-Net baseada na rede VGG-13, com uma arquitetura codificador-decodificador e conexão de salto, tem 23 camadas convolucionais e deconvolucionais, sem nenhuma camada totalmente conectada, como mostra a [Figura 2.72](#).



**Figura 2.72** – Arquitetura codificador-decodificador da rede U-Net com conexão de salto.

## 2.4.4 Modelo DeconvNet

A arquitetura da rede DeconvNet [NOH *et al.* 2015] é composta de duas partes – rede de convolução e rede de deconvolução (Figura 2.73).

A rede de convolução é baseada na rede VGG-16 sem as camadas totalmente conectadas, portanto, tem 13 camadas convolucionais com convoluções  $3 \times 3$  seguidas de operações ReLU e algumas camadas de subamostragem (*pooling*) intermediárias. Duas camadas totalmente conectadas (FC6 e FC7) são adicionadas no final [NOH *et al.* 2015].

Em seguida, uma rede de deconvolução multicamadas gera o mapa de segmentação de uma proposta de entrada. A rede de deconvolução é uma versão espelhada da rede de convolução. Assim, a rede de convolução corresponde ao extrator de atributos que transforma a imagem de entrada em representação de atributos multidimensionais, enquanto a rede de deconvolução prevê o mapa de rótulos de classe a nível de *pixel* e produz a segmentação do objeto a partir do atributo extraído pela rede de convolução. A saída final da rede é um mapa de probabilidade do mesmo tamanho da imagem de entrada, indicando a probabilidade de cada *pixel* pertencer a uma das classes predefinidas.

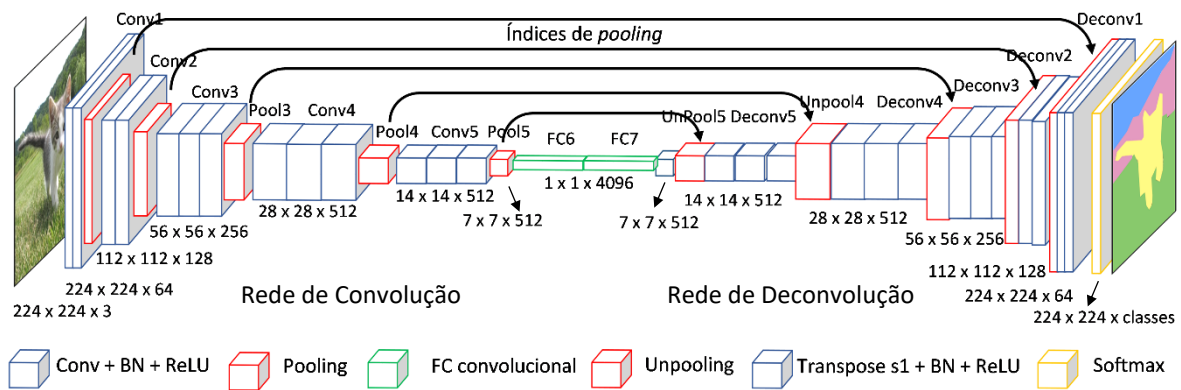


Figura 2.73 – Mapas de atributos da rede DeconvNet. Adaptado de [NOH *et al.* 2015].

Ao contrário da rede de convolução que reduz o tamanho espacial dos mapas de atributos, a rede de deconvolução amplia o tamanho por meio da combinação de operações de *unpooling* e deconvolução:

- (1) *Unpooling*: A subamostragem na rede de convolução é projetada para filtrar ativações ruidosas, abstraindo ativações em um campo receptivo com um único valor representativo. Embora ajude a classificação, retendo apenas ativações robustas nas camadas superiores, as informações espaciais dentro de um campo receptivo são perdidas durante a subamostragem, o que pode ser crítico para obter localização precisa necessária para segmentação semântica. Para resolver esse problema, camadas de *unpooling* são empregadas na rede de deconvolução, realizando a operação reversa do *pooling* e reconstruindo o tamanho original dos mapas de ativações. Para implementar a operação de *unpooling*, os locais de ativações máximas numa janela  $2 \times 2$ , selecionadas durante a operação de *maxpooling* com passo 2, são registrados em variáveis (índices de *pooling*) e empregados nas camadas de *unpooling* correspondentes para colocar cada ativação de volta ao seu local original, conforme ilustrado na Figura 2.74a [NOH *et al.* 2015].

(2) *Deconvolução*: A saída de uma camada de *unpooling* é um mapa de ativação ampliado, mas esparsos (Figura 2.74a). As camadas de deconvolução densificam as ativações esparsas obtidas pelo *unpooling* por meio de operações do tipo convolução transposta  $3 \times 3$  com  $s = 1$  (sem sobreamostragem) com vários filtros aprendidos (Figura 2.74b). Assim, a saída da camada deconvolucional é um mapa de ativação ampliado e denso. Ao contrário das camadas convolucionais, que conectam múltiplas ativações de entrada dentro de uma janela de filtro a uma única ativação, as camadas deconvolucionais associam uma única ativação de entrada com múltiplas saídas, conforme ilustrado na Figura 2.74b. Contudo, na Figura 2.74c, a convolução transposta sem sobreamostragem é implementada como uma convolução  $3 \times 3$  normal, como discutido no Apêndice D.2 [NOH et al. 2015].

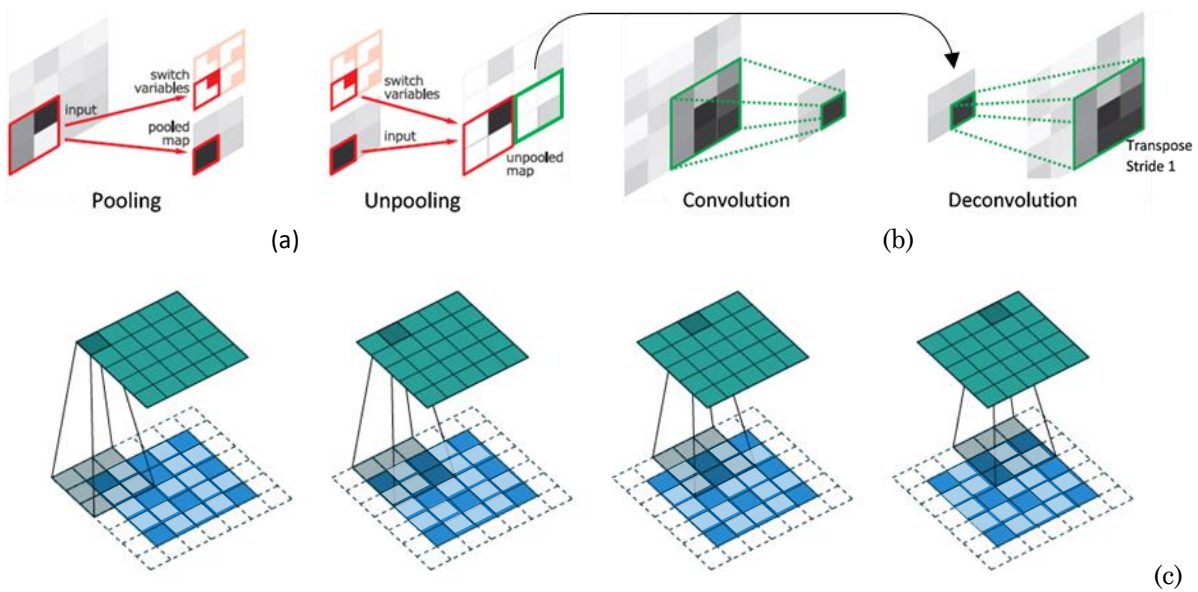


Figura 2.74 – Operações de *unpooling* (a) e *deconvolution* (b) [NOH et al. 2015]. *Convolução transposta de mapa esparsos (azul) com kernel  $3 \times 3$  e  $s = 1$*  (c) [DUMOULIN e VISIN 2016].

A Figura 2.75 mostra a configuração simétrica da arquitetura de convolução e deconvolução, detalhada na Tabela 2.3, centrada em torno da segunda camada totalmente conectada (FC7). As camadas de entrada e saída correspondem à imagem de entrada e aos  $C = 21$  mapas de probabilidade condicional de 21 classes PASCAL-VOC, respectivamente.

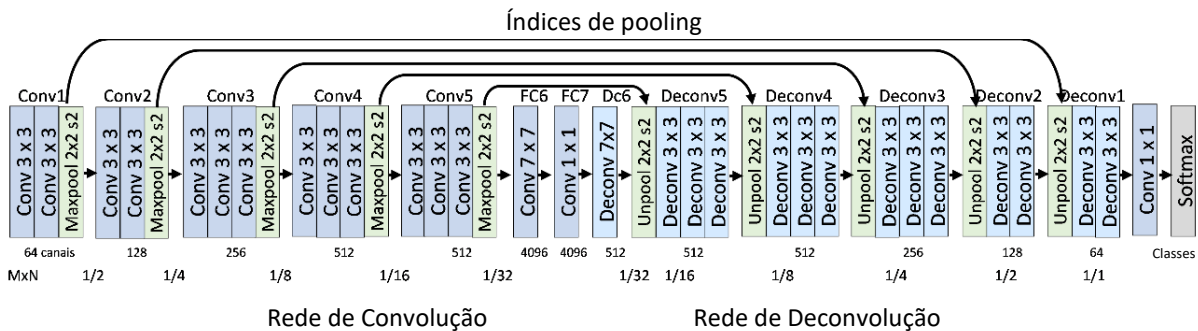


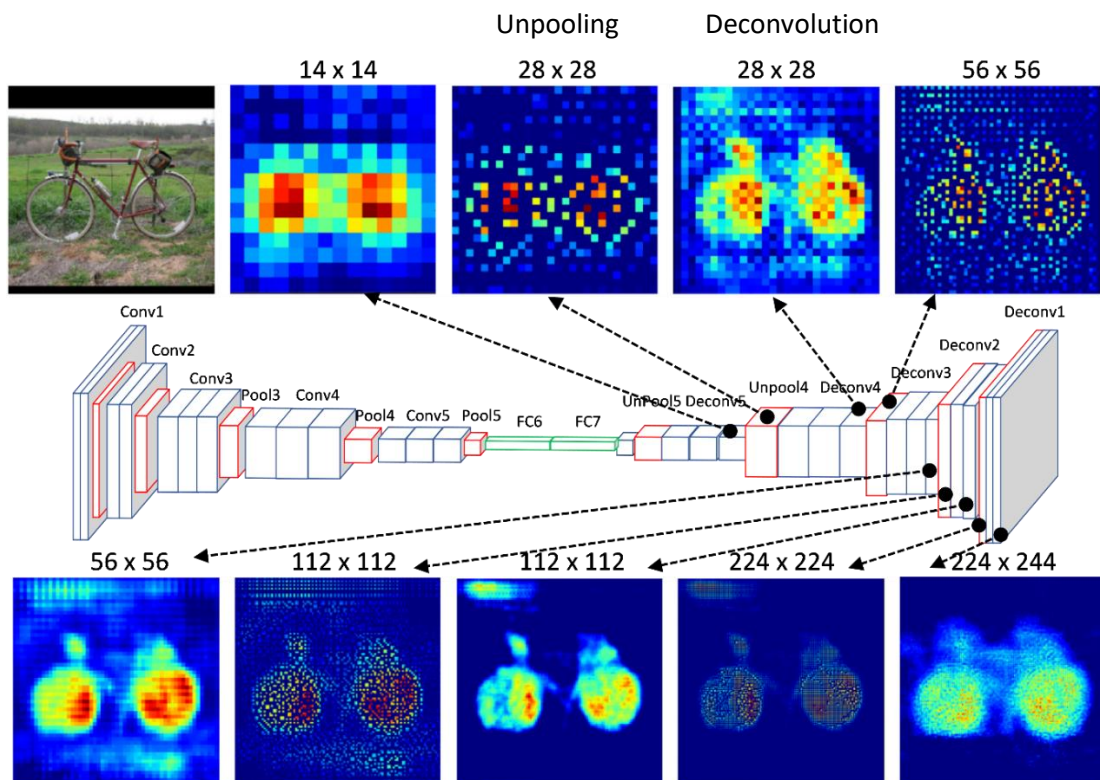
Figura 2.75 – Arquitetura *DeconvNet*: redes de convolução e deconvolução. Baseado em [NOH et al. 2015].

**Tabela 2.3** – Configuração da rede DeconvNet [NOH et al. 2015].

name	kernel size	stride	pad	output size	name	kernel size	stride	pad	output size
input	-	-	-	$224 \times 224 \times 3$	deconv-fc6	$7 \times 7$	1	0	$7 \times 7 \times 512$
conv1-1	$3 \times 3$	1	1	$224 \times 224 \times 64$	unpool5	$2 \times 2$	2	0	$14 \times 14 \times 512$
conv1-2	$3 \times 3$	1	1	$224 \times 224 \times 64$	deconv5-1	$3 \times 3$	1	1	$14 \times 14 \times 512$
pool1	$2 \times 2$	2	0	$112 \times 112 \times 64$	deconv5-2	$3 \times 3$	1	1	$14 \times 14 \times 512$
conv2-1	$3 \times 3$	1	1	$112 \times 112 \times 128$	deconv5-3	$3 \times 3$	1	1	$14 \times 14 \times 512$
conv2-2	$3 \times 3$	1	1	$112 \times 112 \times 128$	unpool4	$2 \times 2$	2	0	$28 \times 28 \times 512$
pool2	$2 \times 2$	2	0	$56 \times 56 \times 128$	deconv4-1	$3 \times 3$	1	1	$28 \times 28 \times 512$
conv3-1	$3 \times 3$	1	1	$56 \times 56 \times 256$	deconv4-2	$3 \times 3$	1	1	$28 \times 28 \times 512$
conv3-2	$3 \times 3$	1	1	$56 \times 56 \times 256$	deconv4-3	$3 \times 3$	1	1	$28 \times 28 \times 256$
conv3-3	$3 \times 3$	1	1	$56 \times 56 \times 256$	unpool3	$2 \times 2$	2	0	$56 \times 56 \times 256$
pool3	$2 \times 2$	2	0	$28 \times 28 \times 256$	deconv3-1	$3 \times 3$	1	1	$56 \times 56 \times 256$
conv4-1	$3 \times 3$	1	1	$28 \times 28 \times 512$	deconv3-2	$3 \times 3$	1	1	$56 \times 56 \times 256$
conv4-2	$3 \times 3$	1	1	$28 \times 28 \times 512$	deconv3-3	$3 \times 3$	1	1	$56 \times 56 \times 128$
conv4-3	$3 \times 3$	1	1	$28 \times 28 \times 512$	unpool2	$2 \times 2$	2	0	$112 \times 112 \times 128$
pool4	$2 \times 2$	2	0	$14 \times 14 \times 512$	deconv2-1	$3 \times 3$	1	1	$112 \times 112 \times 128$
conv5-1	$3 \times 3$	1	1	$14 \times 14 \times 512$	deconv2-2	$3 \times 3$	1	1	$112 \times 112 \times 64$
conv5-2	$3 \times 3$	1	1	$14 \times 14 \times 512$	unpool1	$2 \times 2$	2	0	$224 \times 224 \times 64$
conv5-3	$3 \times 3$	1	1	$14 \times 14 \times 512$	deconv1-1	$3 \times 3$	1	1	$224 \times 224 \times 64$
pool5	$2 \times 2$	2	0	$7 \times 7 \times 512$	deconv1-2	$3 \times 3$	1	1	$224 \times 224 \times 64$
fc6	$7 \times 7$	1	0	$1 \times 1 \times 4096$	output	$1 \times 1$	1	1	$224 \times 224 \times 21$
fc7	$1 \times 1$	1	0	$1 \times 1 \times 4096$					

A Figura 2.76 mostra algumas saídas de camadas de *unpooling* e deconvolucionais, que desempenham papéis diferentes na segmentação. O *unpooling* captura estruturas específicas, recuperando os locais originais das ativações fortes e reconstruindo a estrutura de um objeto em resoluções ampliadas e esparsas. Por outro lado, os filtros em camadas deconvolucionais aprendem a reconstruir a forma de um objeto em diferentes níveis de detalhes, densificando o mapa esparso na saída de *unpooling* [NOH et al. 2015].

A rede DeconvNet demonstrou excelente desempenho no conjunto de dados PASCAL-VOC 2012 [NOH et al. 2015]. No entanto, um conjunto (*ensemble*) de duas redes complementares DeconvNet e FCN-8s foi usado para melhorar o desempenho, calculando a média dos mapas de saída para obter a segmentação semântica final.

**Figura 2.76** – Visualização dos mapas da rede de deconvolução [NOH et al. 2015].



### 2.4.5 Modelo SegNet

As redes de segmentação semântica SegNet, criadas independentemente da rede DeconvNet [NOH *et al.* 2015], consistem em um codificador (*encoder*) com um decodificador correspondente (*decoder*) para mapear os mapas de atributos de baixa resolução na saída do codificador para mapas de atributos de resolução total da entrada, seguida por uma camada final de classificação a nível de *pixel*. As Figuras 2.77 e 2.78 mostram duas versões da rede SegNet [BADRINARAYANAN *et al.* 2015; BADRINARAYANAN *et al.* 2017].

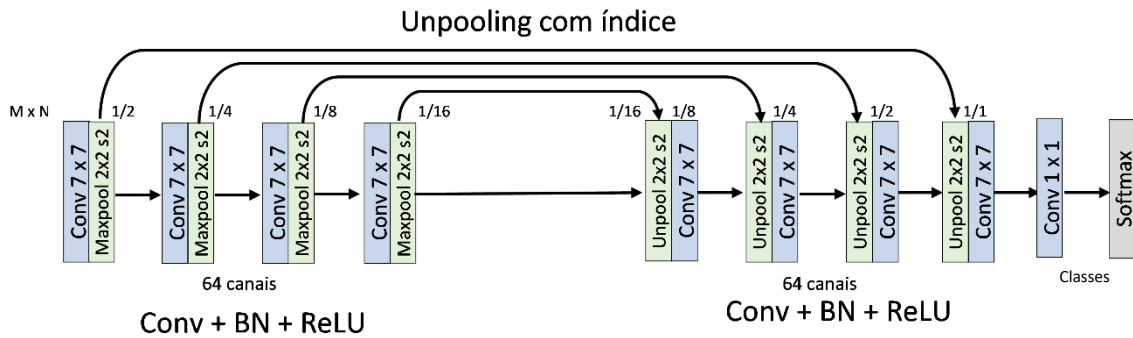


Figura 2.77 – Arquitetura da rede SegNet-v1. Baseado em [BADRINARAYANAN *et al.* 2015].

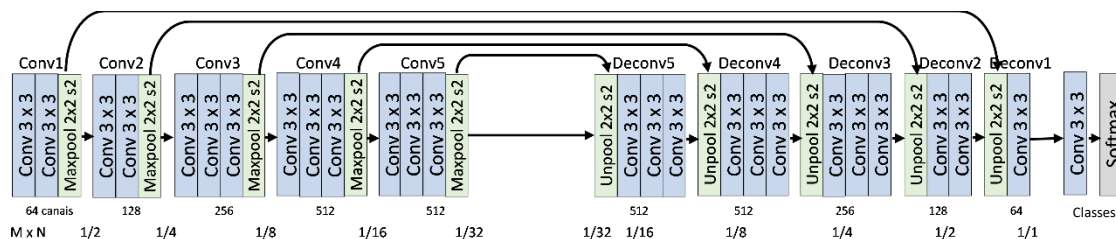


Figura 2.78 – Arquitetura da rede SegNet-v2. Baseado em [BADRINARAYANAN *et al.* 2017].

A primeira versão de SegNet – com quatro camadas de convolução com *kernel*  $7 \times 7$  e camadas de *maxpooling* não sobrepostos  $2 \times 2$  – tem um contexto espacial de  $106 \times 106$  *pixels*. Em outras palavras, um *pixel* no mapa de atributos da camada mais profunda do codificador corresponde a um campo receptivo na imagem de entrada com contexto espacial de  $106 \times 106$  *pixels*, como mostra a Figura 2.79. Usar *kernels* de convolução menores diminuem o contexto e maiores destroem estruturas finas. A adição de mais pares codificador-decodificador na rede SegNet-v2 resulta em um contexto espacial aumentado. As previsões ficam mais suaves à medida que a profundidade aumenta, como consequência de um contexto espacial maior no espaço de entrada [BADRINARAYANAN *et al.* 2015].

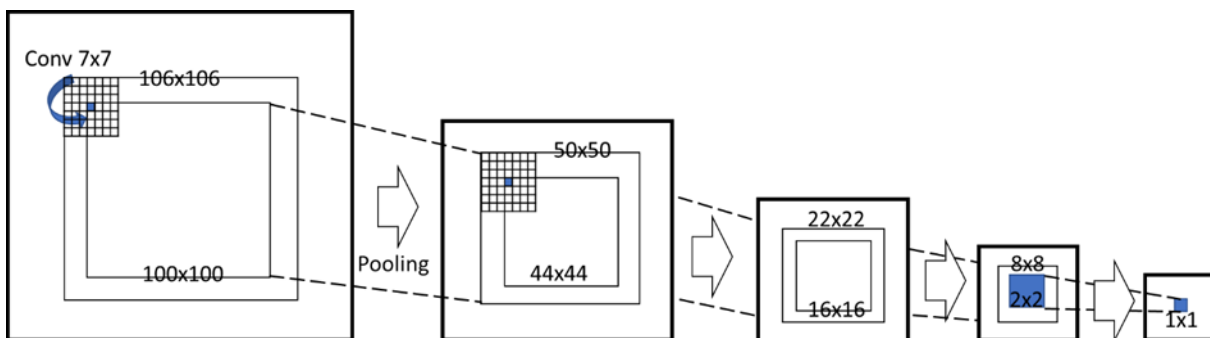
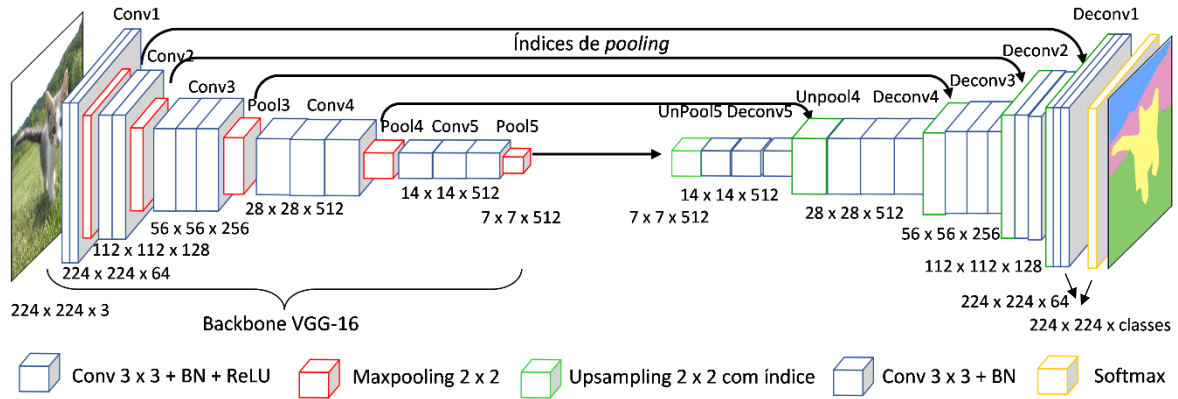


Figura 2.79 – Contexto espacial de SegNet-v1 com campo receptivo de  $106 \times 106$  *pixels*.

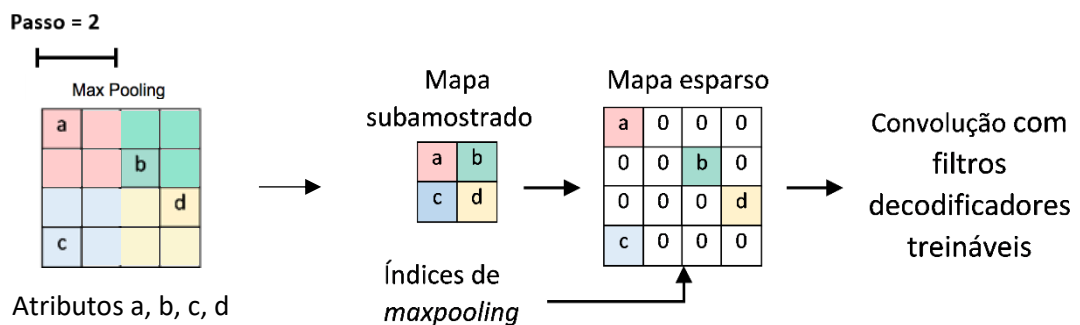
A arquitetura da rede SegNet-v2 é baseada na rede VGG-16, como ilustrada na Figura 2.78. A Figura 2.80 mostra uma representação dos mapas de atributos gerados em cada camada. A rede do codificador é idêntica às primeiras 13 camadas convolucionais de VGG-16 e cada camada de codificador tem uma camada de decodificador correspondente [BADRINARAYANAN *et al.* 2017].



**Figura 2.80** – Mapas de atributos da rede SegNet-v2 com backbone VGG-16 no codificador e um decodificador correspondente. Adaptado de [BADRINARAYANAN *et al.* 2017].

Na Figura 2.80, cada camada de convolução do codificador realiza uma convolução com um banco de filtros, aplica normalização de lote (BN) e uma função de ativação não linear ReLU para produzir um conjunto de mapas de atributos (caixas azuis). Em seguida, uma camada de *maxpooling*  $2 \times 2$  e passo 2 (não sobreposta) faz uma subamostragem dos mapas de atributos por um fator de 2, com perda de resolução espacial (caixas vermelhas). Um tamanho pequeno da janela de *maxpooling* preserva estruturas finas na cena.

As subamostragens das representações da imagem no codificador não são benéficas à segmentação semântica, onde as localizações dos detalhes do objeto são necessárias. Portanto, o codificador deve capturar e armazenar as informações de localização dos atributos antes de realizar a subamostragem. Desta forma, assim como em DeconvNet [NOH *et al.* 2015], os índices das localizações dos máximos calculados na etapa de *maxpooling* são armazenados e passados ao decodificador correspondente. O decodificador usa os índices de *pooling* para fazer o *upsampling*  $2 \times 2$  (sem sobreposição) dos mapas de atributos de entrada, produzindo mapas de atributos esparsos, como mostra a Figura 2.81. O uso de índices melhora a delimitação dos detalhes dos objetos e elimina a necessidade de aprender a fazer *upsampling*.



**Figura 2.81** – Índices de *maxpooling* para *upsampling* (sem aprendizado) dos mapas de atributos. Adaptado de [BADRINARAYANAN *et al.* 2017].

Diferentemente de U-Net que faz uma operação *up-convolution*  $2 \times 2$  (*upsampling* com índices seguida de convolução  $2 \times 2$  para diminuir a quantidade de canais pela metade), DeconvNet e SegNet usam apenas a operação de *upsampling*  $2 \times 2$  com passo 2 usando índices de *maxpooling*. Esta etapa de *upsampling* produz mapas de atributos ampliados, mas esparsos (caixas verdes na [Figura 2.80](#)). Em seguida, o decodificador executa uma sequência de convolução destes mapas com um banco de filtros decodificadores treináveis para produzir mapas de atributos densos (caixas azuis no decodificador). Nas redes SegNet, convoluções  $3 \times 3$  normais são realizadas após o *upsampling* (não treinável) para densificar mapas de atributos, se diferenciando de FCN, onde o *upsampling* treinável é realizado pela convolução transposta  $3 \times 3$  com sobreamostragem ( $s > 1$ ); e de DeconvNet, onde a densificação é realizada por convoluções transpostas  $3 \times 3$  sem sobreamostragem ( $s = 1$ ). Nenhuma não linearidade ReLU é usada nas convoluções do decodificador de SegNet.

Os mapas de atributos de alta dimensão na saída do decodificador são enviados para um classificador multiclasse *softmax* treinável para produzir mapas de probabilidades de classe para cada *pixel* [[BADRINARAYANAN et al. 2017](#)]. Assim, a entrada do *softmax* (caixa laranja) é um mapa de  $C$  canais (número de classes) e a segmentação semântica prevista corresponde à classe com probabilidade máxima em cada *pixel*.

A rede codificadora-decodificadora SegNet pode ser treinada de ponta a ponta em uma tarefa de aprendizado supervisionado, otimizando todos os pesos da rede em conjunto, usando o algoritmo de gradiente descendente estocástico (SGD) e *backpropagation*.

### Considerações sobre as arquiteturas anteriores

As arquiteturas de redes de segmentação semântica estruturadas como codificador-decodificador, como DeconvNet e SegNet-v2, têm redes codificadoras idênticas às camadas convolucionais da rede VGG-16 e, portanto, o codificador normalmente usa as cinco camadas convolucionais pré-treinadas no conjunto de dados de classificação ImageNet (diferentemente da rede codificadora da U-Net original, baseada na rede VGG-13, projetada especificamente para segmentação de imagens biomédicas). No entanto, as redes DeconvNet e SegNet-v2 diferem na rede decodificadora, treinamento e inferência.

Comparando a rede DeconvNet na [Figura 2.75](#) com a rede SegNet-v2 na [Figura 2.78](#), as camadas internas FC6 (conv  $7 \times 7$ ), FC7 (conv  $1 \times 1$ ) e Deconv6 (deconv  $7 \times 7$ ) desaparecem, portanto, a SegNet não tem camadas totalmente conectadas, sendo uma rede totalmente convolucional. Assim, ao contrário de outras arquiteturas, como FCN e DeconvNet, a rede SegNet não usa as camadas FC convolucionalizadas entre o codificador e decodificador, com intuito de preservar mapas de atributos de resolução mais alta na saída do codificador mais profundo. Isso reduz o número de parâmetros treináveis da rede-base VGG-16 no codificador, de 134M para 14,7M, aumentando a velocidade de treinamento e inferência. Portanto, a SegNet é significativamente menor e mais fácil de treinar do que outras arquiteturas que utilizam camadas FC convolucionalizadas, que podem ter centenas de milhões de parâmetros treináveis [[BADRINARAYANAN et al. 2017](#)].

Por exemplo, FCN é uma arquitetura que possui um grande número de parâmetros treináveis na rede do codificador com camadas totalmente convolucionais (134M), mas uma rede decodificadora muito pequena (0,5M). A dificuldade de treinar essa rede levou

ao treinamento em múltiplos estágios, anexando gradualmente camadas de deconvolução nas redes FCN pré-treinadas até obter a rede FCN-8s.

O codificador de DeconvNet contém as camadas totalmente conectadas FC6 e FC7 da rede VGG-16, que corresponde a cerca de 90% dos parâmetros de toda a rede. Logo, DeconvNet é duas vezes mais profunda e complexa que VGG-16 e contém muitos parâmetros treináveis, dificultando o treinamento em conjuntos de dados menores. Por este motivo, DeconvNet usa um método de treinamento de dois estágios, onde a rede é treinada primeiro com exemplos fáceis e depois é ajustada com propostas de região mais desafiadoras.

Para capturar e armazenar informações de detalhes nos mapas de atributos intermediários do codificador, as arquiteturas com conexões de salto, como FCN e U-Net, necessitam de mais recursos de memória para armazenar e transferir mapas de atributos mais refinados do codificador, para adicionar (no caso de FCN) ou concatenar (no caso de U-Net) com mapas de atributos mais grosseiros na saída da rede. Por outro lado, as redes codificador-decodificador sem conexão de salto, como DeconvNet e SegNet, armazenam e transferem apenas os índices de *maxpooling* aos decodificadores correspondentes, ou seja, usam apenas 2 bits para armazenar as localizações do valor de atributos máximo em cada janela  $2 \times 2$  dos mapas do codificador [BADRINARAYANAN *et al.* 2017].

#### 2.4.6 Modelo RefineNet

Como mostrado na Figura 2.55, múltiplas operações de subamostragem em CNNs profundas levam a uma diminuição significativa na resolução inicial da imagem por um fator de 32 em cada dimensão. Para gerar previsões semânticas de alta resolução, os modelos de rede codificador-decodificador, como DeconvNet [NOH *et al.* 2015] e SegNet-v2 [BADRINARAYANAN *et al.* 2017], aprendem camadas de deconvolução, aumentando a amostragem das previsões de baixa resolução. No entanto, as operações de deconvolução não são capazes de recuperar os atributos visuais de baixo nível (das camadas rasas) que são perdidos após as operações de subamostragem no codificador e, portanto, são incapazes de produzir uma previsão precisa de alta resolução com os limites e detalhes de objetos.

As arquiteturas de redes com conexões de salto, por exemplo, Hypercolumns [HARIHARAN *et al.* 2014a], FCN [LONG *et al.* 2015] e U-Net [RONNEBERGER *et al.* 2015], exploram atributos de camadas intermediárias, que retêm informações espaciais para gerar previsão de alta resolução. As informações das camadas intermediárias são complementares aos atributos das camadas iniciais de convolução (que codificam informações visuais espaciais de baixo nível, como bordas e cantos), e também aos atributos de alto nível de camadas mais profundas (que codificam informações semânticas mais abstratas). Os atributos de todos os níveis são úteis para uma previsão da segmentação semântica com precisão, uma vez que atributos semânticos de alto nível ajudam no reconhecimento de classes de regiões da imagem, enquanto atributos visuais de baixo nível ajudam a gerar limites nítidos e detalhados para previsão de alta resolução.

Na Figura 2.82, a rede RefineNet [LIN, MILAN *et al.* 2017] emprega como *backbone* as redes residuais de classificação ResNet-50, 101 ou 152 [HE *et al.* 2016a] com conexões residuais de curto alcance dos mapeamentos de identidade. A rede RefineNet explora atributos de vários níveis usando conexões de longo alcance (conexões de salto ou *skip connections*) para gerar previsões de alta resolução, de modo que os gradientes podem ser



propagados diretamente através de conexões de curto e longo alcance, permitindo treinamento de ponta a ponta [DROZDZAL *et al.* 2016]. Esta rede refina atributos semânticos de alto nível de baixa resolução (grosseiros) com atributos de baixo nível refinados, para gerar mapas de atributos semânticos de maior resolução.

Comparando a Figura 2.71 e Figura 2.82, podemos perceber que a rede RefineNet segue a mesma estrutura básica da rede U-Net, porém com módulos de refinamento mais complexos e convoluções residuais. Para efeito ilustrativo, redesenhamos a rede RefineNet na Figura B.4 no Apêndice B, para comparação de sua arquitetura com a rede FCN-8s (adição) na Figura 2.67 e a rede U-Net (concatenação) na Figura 2.72, mostrando as similaridades estruturais entre elas.

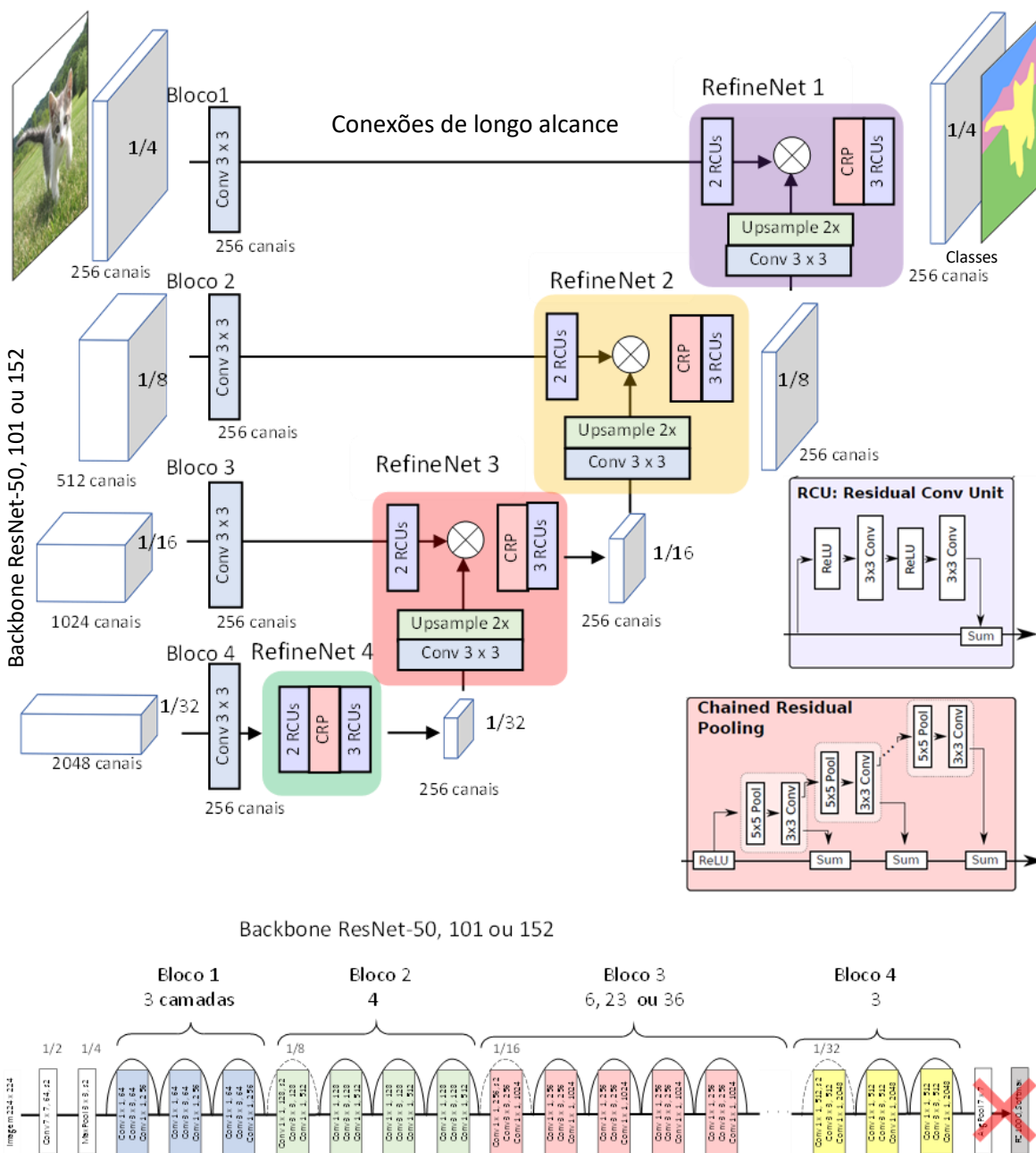


Figura 2.82 – Arquitetura de rede RefineNet. Adaptado de [LIN, MILAN *et al.* 2017].

Primeiro, cada saída dos quatro blocos ResNet passa por uma camada convolucional  $3 \times 3$  para adaptar a dimensionalidade (número de canais). Começando a partir do último bloco da ResNet, a saída do bloco 4 de ResNet é conectada ao bloco RefineNet-4. No próximo estágio, a saída do RefineNet-4 e do bloco 3 são alimentados no RefineNet-3. Da mesma forma, o RefineNet-2 e o RefineNet-1 repetem esse refinamento em etapas, fundindo informações de alto nível das camadas profundas da CNN com os atributos de alta resolução de baixo nível das primeiras camadas [LIN, MILAN *et al.* 2017]. Para isso, foram introduzidas conexões residuais de longo alcance entre as saídas dos quatro blocos ResNet e os módulos RefineNets correspondentes. Essas conexões transmitem os atributos finos de baixo nível das camadas rasas (que codificam detalhes visuais), com o propósito de refinar os mapas de atributos grosseiros de alto nível das camadas profundas.

O bloco RefineNet consiste em componentes que são capazes de refinar os atributos semânticos grosseiros de alto nível, explorando atributos visuais de baixo nível. Os componentes de convolução residual (*Residual Conv Unit* – RCU) ajustam os pesos pré-treinados da ResNet para uma nova tarefa. Cada caminho de entrada é passado sequencialmente através de duas unidades RCU.

Além disso, um componente chamado *Chained Residual Pooling* – CRP captura o contexto de fundo de uma região da imagem. Para isso, os atributos são agrupados sem subamostragem (*maxpooling*  $5 \times 5$  com  $s = 1$ ) e, em seguida, passam por uma convolução  $3 \times 3$ . Estas operações são repetidas dentro de um encadeamento, onde o bloco de *pooling* atual reutiliza o resultado do *pooling* anterior e, assim, acessa os atributos de uma grande região sem usar uma janela grande de *pooling*. Os mapas de atributos de saída de todos os blocos de *pooling* são fundidos por adição com o mapa de atributos de entrada com conexões residuais, como mostrado na [Figura 2.82](#) (caixa *Chained Residual Pooling*).

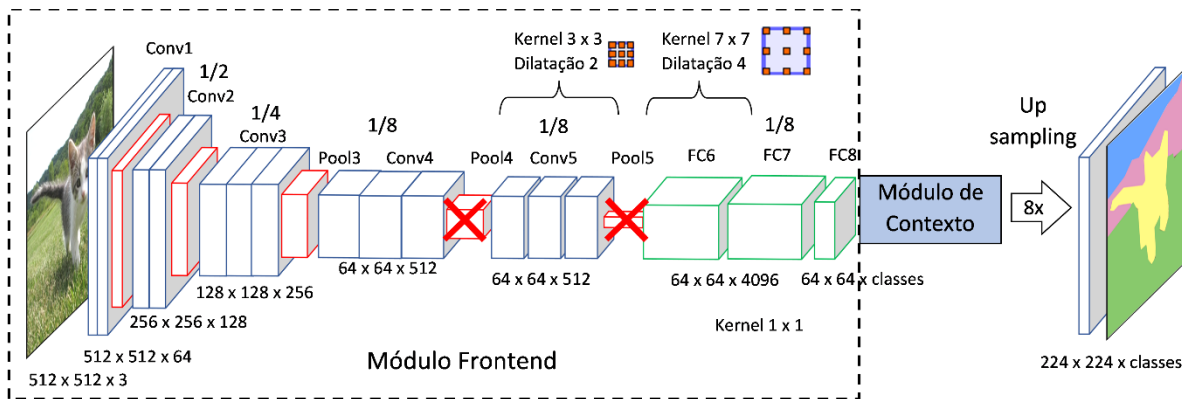
Como última etapa, os mapas de atributos finais de alta resolução são alimentados em uma camada *softmax* densa, com objetivo de gerar a previsão final na forma de um mapa de pontuação denso. Este mapa de pontuação é, então, sobreamostrado com interpolação bilinear de 4x para recuperar a resolução da imagem original de entrada.

### 2.4.7 Modelo DilatedNet

As CNNs integram informações contextuais mais abstratas por meio de camadas sucessivas de subamostragem que reduzem a resolução até obter uma previsão de contexto global. No entanto, a segmentação semântica requer informação de contexto concomitante à previsão densa em nível de *pixel* na mesma resolução da entrada. Ao contrário das arquiteturas em forma de pirâmide herdadas da classificação de imagens, um módulo de contexto pode ser conectado às arquiteturas de segmentação existentes, com a intenção de agregar informações contextuais sem perder resolução.

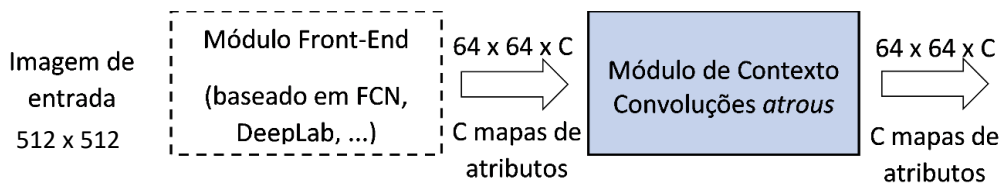
Na rede DilatedNet [YU e KOLTUN 2016], um módulo de contexto foi anexado a uma rede *frontend* para obter previsão densa, como mostrado na [Figura 2.83](#). O *frontend* pode ser qualquer rede modificada de segmentação, como FCN-32s [LONG *et al.* 2015] ou DeepLab-v1 [CHEN *et al.* 2015], descrita posteriormente na [Seção 2.4.8](#). Seguindo os mesmos conceitos da rede FCN, o *backbone* VGG-16 foi convolucionalizado e adaptado para previsão densa, entretanto, as duas últimas camadas de subamostragem (*pooling*) da CNN foram removidas para produzir uma saída de maior resolução com fator de 8 (em vez de

32). Outra diferença é que as convoluções  $3 \times 3$  nas três camadas convolucionais de Conv5 foram dilatadas por um fator de 2; e a convolução  $7 \times 7$  em FC6, por um fator de 4.



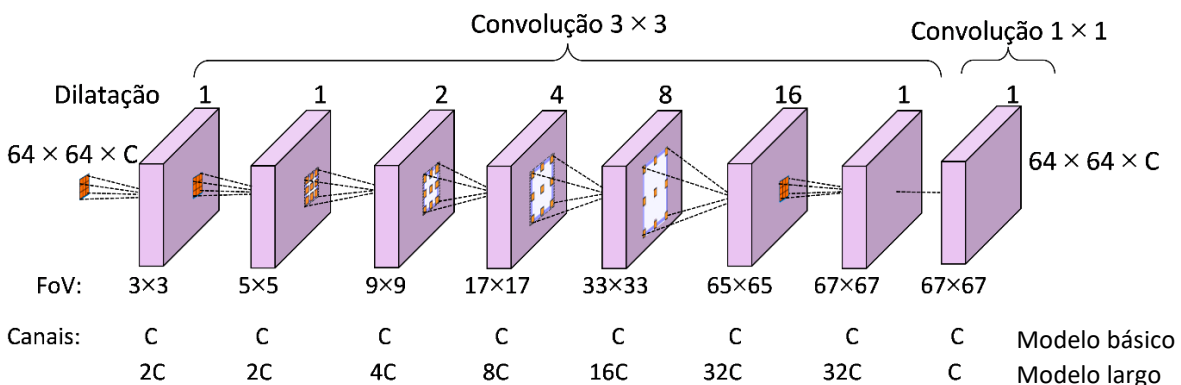
**Figura 2.83** – Mapas de atributos do módulo frontend FCN-32s (VGG-16) modificado, seguido do módulo de contexto da rede DilatedNet.

A Figura 2.84 mostra que o módulo frontend recebe uma imagem RGB de  $512 \times 512$  como entrada e produz  $C$  mapas de atributos de  $64 \times 64$  como saída ( $C = 21$  classes PASCAL-VOC 2012 e  $M/8 \times N/8 = 64 \times 64$ ). Em seguida, o módulo de contexto recebe  $C$  mapas de atributos como entrada e produz  $C$  mapas de mesmo tamanho na saída. Portanto, este módulo pode ser conectado a qualquer rede de previsão densa existente.



**Figura 2.84** – Módulo frontend com uma FCN modificada (FCN-32s ou DeepLab-v1), seguido de um módulo de contexto.

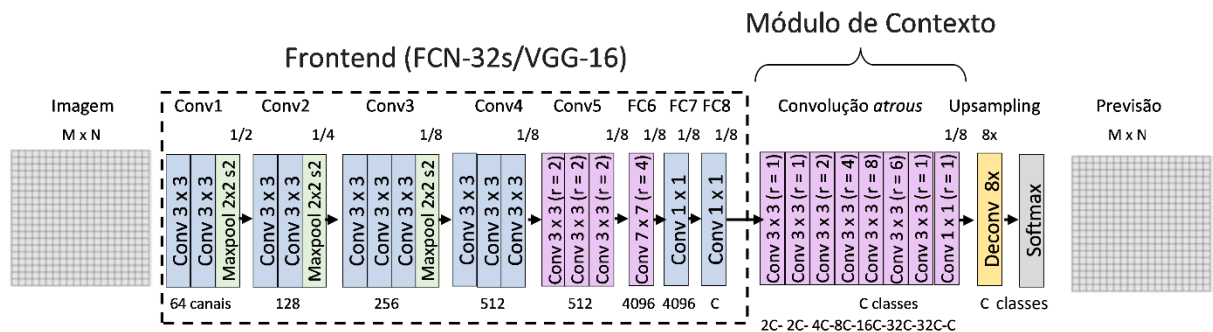
O módulo de contexto possui apenas camadas convolucionais baseadas em convoluções dilatadas (convoluções *atrous*), com a finalidade de agregar o contexto com campos receptivos (ou campo de visão – FoV) crescentes, de  $3 \times 3$  até  $67 \times 67$  pixels, sem usar operações de subamostragem, a fim de manter a resolução espacial dos mapas de atributos na saída, como mostra a Figura 2.85.



**Figura 2.85** – Configuração de um módulo de contexto da rede DilatedNet. Baseado em [YU e KOLTUN 2016].

Na [Figura 2.85](#), o módulo de contexto emprega sete camadas de convoluções  $3 \times 3$  com taxas de dilatação crescentes 1, 1, 2, 4, 8, 16 e 1, que permitem ampliar o campo de visão (FoV) dos filtros para incorporar um contexto maior ( $67 \times 67 \text{ pixels}$ ). No módulo de contexto básico, cada camada possui mapas de atributos com  $C$  canais. Opcionalmente, um modelo largo usa um número crescente de mapas de atributos de  $1C$  na entrada até  $32C$  nas camadas mais profundas, apresentando maior precisão do que o módulo básico. Uma camada final executa convoluções com *kernel*  $1 \times 1 \times C$  e produz a saída do módulo com mesma resolução espacial de  $64 \times 64$  (fator 8) com  $C$  canais. Após o módulo de contexto, o mapa de saída passa por um *upsampling* de 8x para recuperar a resolução original de entrada e por uma camada *softmax* final para gerar os mapas de previsão densa.

A arquitetura da rede DilatedNet é mostrada em detalhes na [Figura 2.86](#). Os experimentos mostraram que o módulo de contexto com convoluções *atrous*, conectado a arquiteturas de segmentação semântica, como FCN-32s adaptada com convolução *atrous*, aumenta a precisão da previsão densa [YU e KOLTUN 2016].



**Figura 2.86** – Arquitetura DilatedNet com frontend FCN-32s modificado e módulo de contexto com convolução *atrous*. Baseado em [YU e KOLTUN 2016].

## Convolução *atrous*

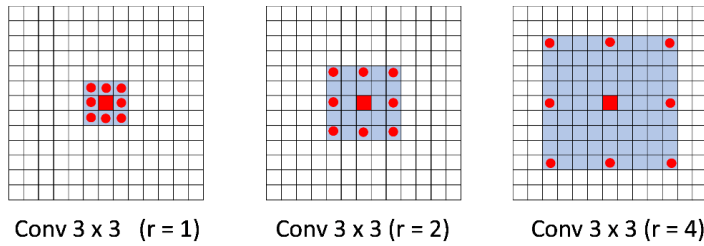
A convolução dilatada ou *atrous* foi originalmente desenvolvida para calcular de forma eficiente a transformada *wavelet* discreta [MALLAT 2008]. Em um caso unidimensional, a convolução *atrous* pode ser formulada da seguinte forma:

$$y[i] = \sum_k x[i + r \cdot k] w[k], \quad (2.5)$$

sendo que  $x[i]$  denota o sinal de entrada,  $y[i]$  denota o sinal de saída,  $r$  é a taxa de dilatação,  $w[k]$  denota o  $k$ -ésimo parâmetro do filtro.

Para cada localização  $i$  na saída  $y$  e um filtro  $w$ , a convolução *atrous* é aplicada sobre o mapa de atributos de entrada  $x$ , onde a taxa de dilatação  $r$  corresponde ao passo de entrada com a qual o sinal de entrada é amostrado. Ajustando  $r$ , é possível modificar o campo de visão do filtro (FoV), onde um valor de taxa grande aumenta o campo de visão para incorporar um contexto maior. A convolução padrão corresponde à equação da convolução *atrous* com taxa  $r = 1$ .

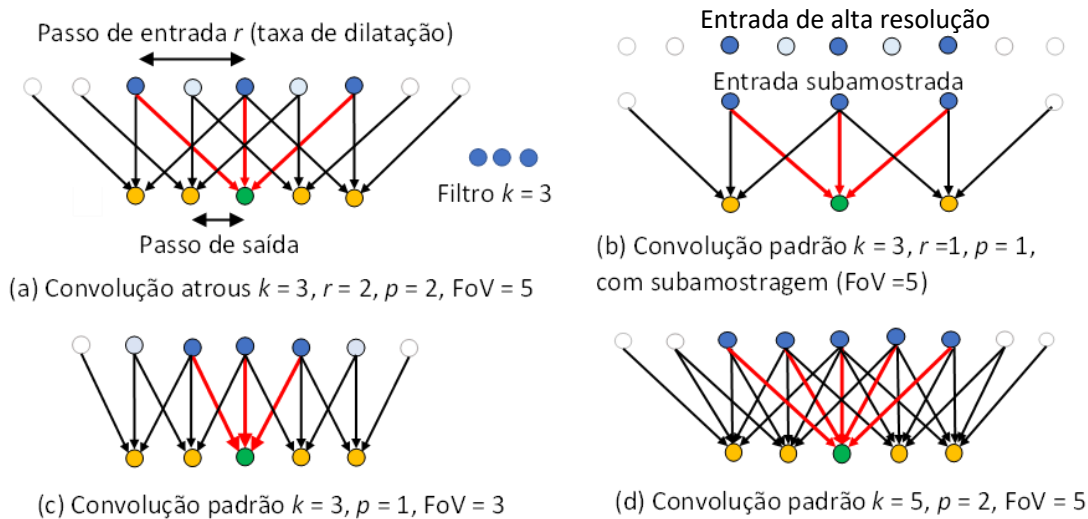
A convolução *atrous* 2D é equivalente à convolução da entrada  $x$  com filtros dilatados com taxa  $r$ , inserindo  $r - 1$  zeros entre dois valores de filtro consecutivos ao longo de cada dimensão espacial, como ilustrado na [Figura 2.87](#).



**Figura 2.87** – *Convolução atrous 2D com kernel  $3 \times 3$  e diferentes taxas de dilatação  $r = 1, 2$  e  $4$ . Pixels vermelhos denotam as entradas ponderadas pelos pesos do kernel, com pixel de saída no centro. Pixels azuis denotam o campo receptivo. Adaptado de [CHEN et al. 2017].*

Para extrair informações de alto nível semântico, as CNNs usam várias camadas de *pooling* para aumentar o tamanho do campo receptivo (FoV) de um neurônio de saída, porém com o inconveniente de reduzir o tamanho do mapa de atributos final, que deve ser sobreamostrado para recuperar a resolução de entrada em aplicações de segmentação semântica. A convolução *atrous* permite manter a resolução do mapa de atributos de entrada de uma determinada camada e simultaneamente aumentar o campo receptivo; por este motivo, foi utilizada na rede DilatedNet e em várias versões da rede DeepLab.

Para exemplificar, a **Figura 2.88** representa convoluções 2D em uma dimensão, por simplicidade. Na **Figura 2.88b**, a convolução padrão  $3 \times 3$  é realizada em um mapa de atributos de baixa resolução (subamostrado após *maxpooling*), gerando um mapa de atributos esparsos de baixa resolução, com campo receptivo correspondente a  $5 \times 5$  na entrada original. Por outro lado, na **Figura 2.88a**, uma convolução *atrous* de *kernel*  $3 \times 3$  com taxa  $r = 2$  (sobre um mapa de atributos de entrada de alta resolução) gera um mapa de atributos denso de mesma resolução. Portanto, a convolução *atrous* permite manter o tamanho do filtro e a resolução espacial de saída, sem diminuir o campo receptivo.



**Figura 2.88** – *Extração de atributos densos com convolução atrous com kernel  $k = 3$ , taxa de dilatação  $r = 2$  e passo de saída  $= 1$ , aplicada em uma entrada de alta resolução com preenchimento  $p = 2$  (a); Extração de atributos esparsos com convolução padrão,  $k = 3$ , passo de entrada e saída igual a 1, em uma entrada subamostrada de baixa resolução (b); Extração de atributos densos com convolução padrão,  $k = 3$ , passo de entrada e saída 1, em uma entrada de alta resolução e menor campo receptivo ( $FoV = 3$ ) (c); Convolução padrão com kernel grande ( $k = 5$ ), passo de entrada e saída 1, com maior número de parâmetros e campo receptivo ( $FoV = 5$ ) (d). Adaptado de [CHEN et al. 2015; CHEN et al. 2018a].*

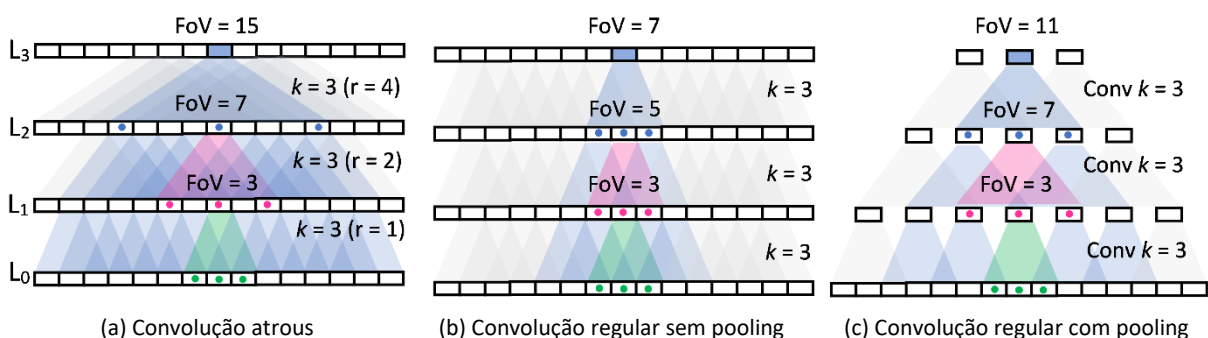


Se a operação de subamostragem fosse eliminada na [Figura 2.88b](#) para manter a resolução da entrada, o campo receptivo da convolução  $3 \times 3$  padrão diminuiria para  $3 \times 3$ , como mostra a [Figura 2.88c](#). Sendo assim, na [Figura 2.88a](#), uma convolução *atrous* sem subamostragem sobre uma entrada de alta resolução permite manter a resolução da entrada, mas cada neurônio de saída possui um campo receptivo maior e, portanto, codifica atributos de maior nível semântico.

Além disso, a convolução *atrous*  $3 \times 3$  com uma taxa  $r = 2$  ([Figura 2.88a](#)) tem o mesmo tamanho de *kernel* de uma convolução padrão  $3 \times 3$  ([Figura 2.88c](#)). Conseqüentemente, a convolução *atrous* permite ampliar efetivamente o campo de visão dos filtros, ajustando a taxa de dilatação (passo de entrada), sem aumentar o número de parâmetros do *kernel* e a quantidade de computação. Por exemplo, a convolução *atrous*  $3 \times 3$  ([Figura 2.88a](#)) tem o mesmo campo de visão que um *kernel* padrão  $5 \times 5$  ([Figura 2.88d](#)), embora use apenas 9 parâmetros em vez de 25.

O módulo de contexto em DilatedNet ([Figura 2.85](#)) é baseado em uma sequência de convoluções com taxas de dilatação exponenciais para expansão do campo receptivo até  $67 \times 67$  sem perda de resolução da entrada. Considerando apenas três camadas convolucionais consecutivas com filtros de tamanho  $k = 3$  e taxas de dilatação exponencialmente crescentes ( $r = 1, 2$  e  $4$ ), na [Figura 2.89a](#), o campo receptivo de um elemento  $p$  no mapa de atributos da camada  $L_{i+1}$  é o conjunto de elementos na camada inicial  $L_0$  que modifica o valor de  $p$ . Portanto, o tamanho do campo receptivo de cada elemento na camada  $L_{i+1}$  também cresce exponencialmente, calculado por  $(2^{i+2} - 1)$  [YU e KOLTUN 2016]. A [Figura 2.89a](#) mostra uma representação 1D de convoluções  $3 \times 3$  com taxas de dilatação exponencialmente crescentes 1, 2 e 4, com a finalidade de produzir os mapas densos das camadas  $L_1$ ,  $L_2$  e  $L_3$ , respectivamente. Cada elemento de  $L_1$ ,  $L_2$  e  $L_3$  tem um campo receptivo FoV de  $3 \times 3$ ,  $7 \times 7$  e  $15 \times 15$ , respectivamente.

Sem operações de *pooling* intercaladas, uma pilha de convoluções com taxas de dilatação crescentes ([Figura 2.89a](#)) geram um FoV maior do que uma pilha de três convoluções regulares ([Figura 2.89b](#)). Desta forma, como o módulo *frontend* de DilatedNet ([Figura 2.83](#)) remove as duas últimas camadas de subamostragem, convoluções *atrous* com taxas 2 e 4 são usadas nas camadas de convolução subsequentes, para obter um campo receptivo maior e manter a resolução ao mesmo tempo. Além disso, a convolução *atrous* permite controlar o tamanho do campo receptivo com taxas variadas no módulo de contexto, mantendo a resolução na saída, sem aumentar o número de parâmetros, em contraste com as convoluções regulares fixas das redes CNN com operações de *maxpooling* intercaladas (mapas de atributos esparsos) ([Figura 2.89c](#)).



**Figura 2.89** – Campos receptivos de uma pilha de convoluções regulares ou atrous.

### 2.4.8 Modelos DeepLab

A primeira versão do modelo DeepLab [CHEN *et al.* 2015] é uma variante de rede totalmente convolucional, que adota a rede VGG-16 como *backbone*, com objetivo de extrair mapas de atributos densos para classificação refinada de *pixels*. Sendo assim, as últimas camadas totalmente conectadas da rede VGG-16 original foram transformadas em camadas convolucionais, como já mostrado na Figura 2.61.

Como vimos na Figura 2.62, o fator de decimação espacial da FCN-32s é 32 devido ao emprego de cinco camadas *maxpooling*, cada uma com passo 2. No entanto, isso não é desejável por produzir previsão muito esparsa. Em contraste com a rede FCN-32s, a rede DeepLab-v1 básico na Figura 2.90 reduz o fator de decimação espacial para 8, descartando a subamostragem nos dois últimos blocos da rede VGG. Ao contrário de DilatedNet, que descarta duas operações de *pooling*, a rede DeepLab-v1 aplica operações de *maxpooling*  $3 \times 3$  sobreposto (com passo 1 e preenchimento 1) em Pool4 e Pool5 para manter a resolução espacial, mantendo o fator de redução de 8x a partir do quarto bloco de convolução (Conv4), o que resulta em previsões mais densas.

Para aumentar o campo receptivo e diminuir a complexidade computacional, as três camadas convolucionais em Conv5 são modificadas para convolução *atrous*  $3 \times 3$  com taxa de dilatação  $r = 2$ , e FC6 aplica uma camada convolucional *atrous*  $4 \times 4$  com taxa  $r = 4$ .

Na Figura 2.62, podemos observar que a rede FCN-32s não usa o algoritmo *atrous* e produz previsões muito grosseiras (subamostradas por um fator de 32) na saída da CNN, sendo necessário usar uma camada de *upsampling* com convolução transposta de 32x para recuperar a resolução original. No entanto, na Figura 2.90, os mapas de pontuação de classe da camada final com fator de 8 do DeepLab são mais suaves, por isso, a rede emprega interpolação bilinear simples de 8x para aumentar sua resolução de saída de  $M/8 \times N/8$  para a resolução da imagem original.

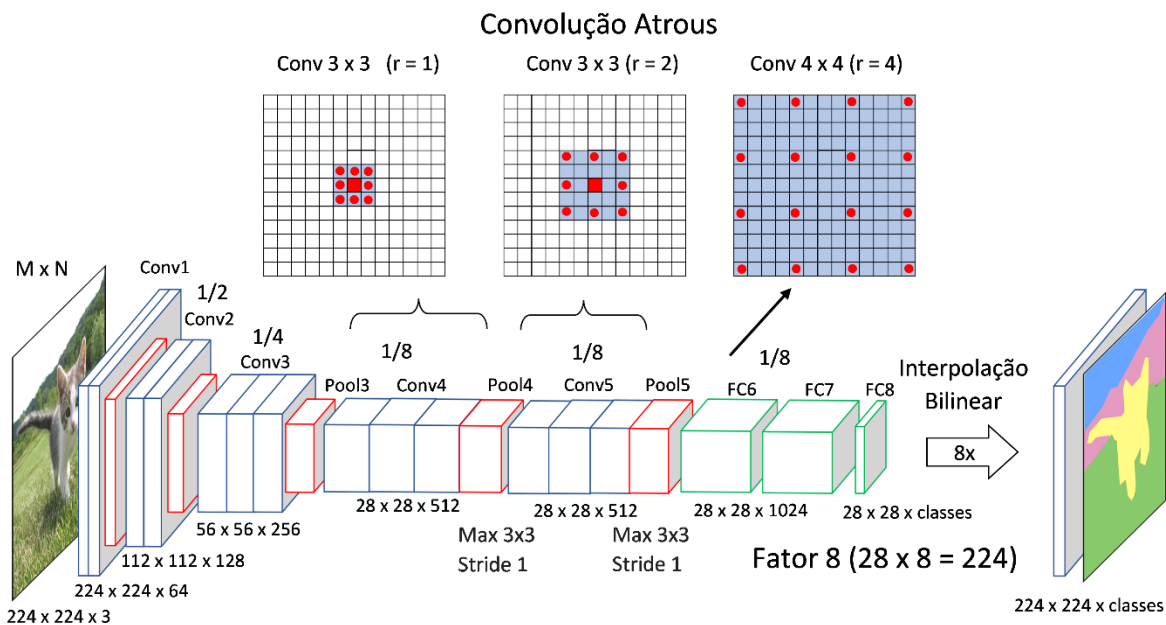
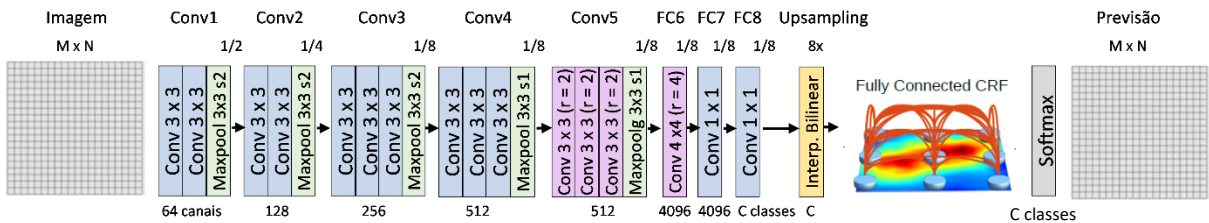


Figura 2.90 – Mapas de atributos da rede DeepLab-v1 básica com convolução atrous.



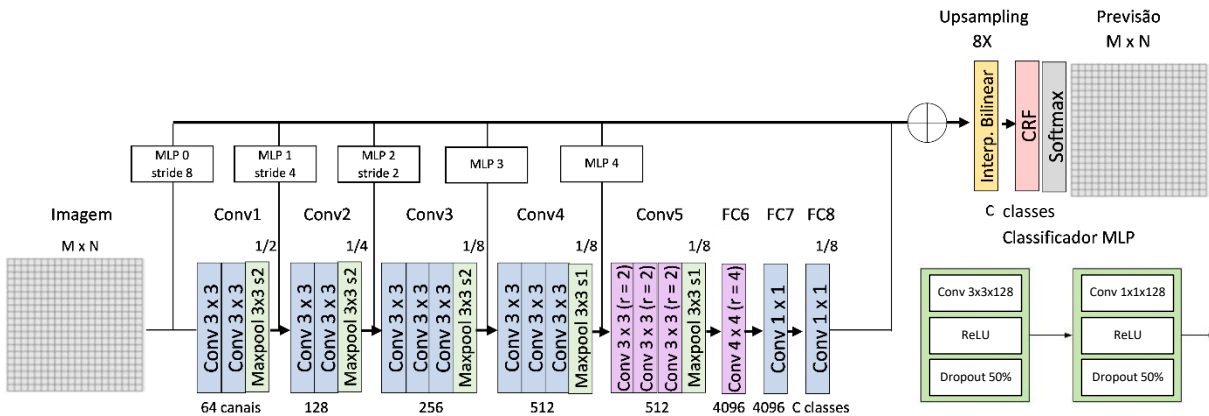
A **Figura 2.91** mostra outras diferenças da estrutura do modelo DeepLab-v1 básico com a rede FCN-32s. A subamostragem nos três primeiros blocos de convolução foi modificado de *maxpooling*  $2 \times 2$  (passo 2 e preenchimento 0) para  $3 \times 3$  (com passo 2 e preenchimento 1) para reduzir o tamanho do mapa de atributos pela metade. Além disso, a capacidade do modelo capturar detalhes finos das bordas do objeto foi aumentada, empregando um método de campo aleatório condicional (*Conditional Random Field – CRF*) totalmente conectado [KRAHENBUHL e KOLTUN 2011]. A saída da última camada sobreamostrada de 8x é usada como entrada para inferência do CRF.



**Figura 2.91** – Arquitetura da rede DeepLab-v1 básica.

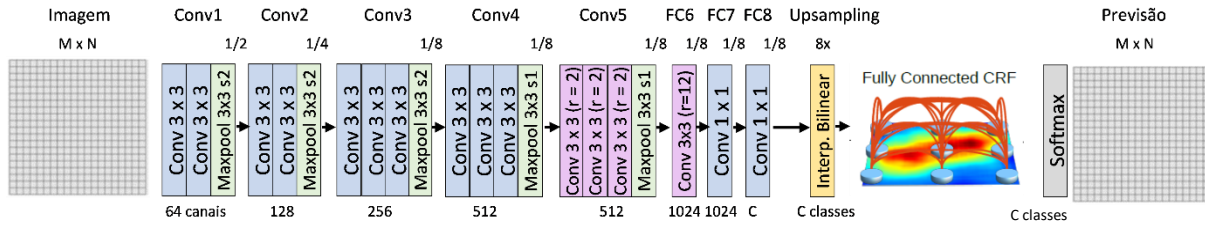
Ao contrário da abordagem deconvolucional (com convolução transposta) adotada por FCN [LONG *et al.* 2015], a rede DeepLab-v1 converte redes de classificação de imagens em extratores de atributos densos sem a necessidade de aprender nenhum parâmetro extra com redes de deconvolução, levando a um treinamento mais rápido. No entanto, além do modelo DeepLab-v1 básico (**Figura 2.91**), foram criadas duas variantes para obter melhor precisão: DeepLab-MSc (**Figura 2.92**) que explora atributos multiescala e DeepLab-LargeFOV (**Figura 2.93**) com grande campo de visão.

Naquela época, seguindo os resultados promissores de previsão de FCN-8s com uma arquitetura de salto que agrega mapas de atributos de camadas convolucionais intermediárias para aumentar a precisão da localização e refinar os limites do objeto, uma variante DeepLab-MSc com conexões de salto foi criada para incorporar atributos multiescala. Para isso, um classificador *Multilayer Perceptron* (MLP) de duas camadas (com 128 filtros convolucionais  $3 \times 3$  na primeira camada e 128 filtros  $1 \times 1$  na segunda camada) foi anexado à imagem de entrada e às saídas das primeiras quatro camadas *maxpooling*. Os mapas de atributos resultantes são concatenados ao mapa de atributos da última camada da rede principal. O mapa de atributos agregado alimentado na camada *softmax* é, portanto, aprimorado com  $5 \times 128 = 640$  canais.



**Figura 2.92** – Arquitetura da rede DeepLab-MSc com backbone VGG e conexões de salto: os blocos brancos são classificadores MLP (detalhe no quadro tracejado lateral).

Na variante DeepLab-LargeFoV mostrada na [Figura 2.93](#), projetada para obter maior campo de visão, a camada FC6 aplica uma convolução *atrous* com um campo receptivo maior usando um filtro *atrous*  $3 \times 3$  com taxa de dilatação  $r = 12$ . Além disso, uma redução do número de canais nas duas camadas FC de 4.096 para 1.024 diminui o tempo de computação e o consumo de memória.



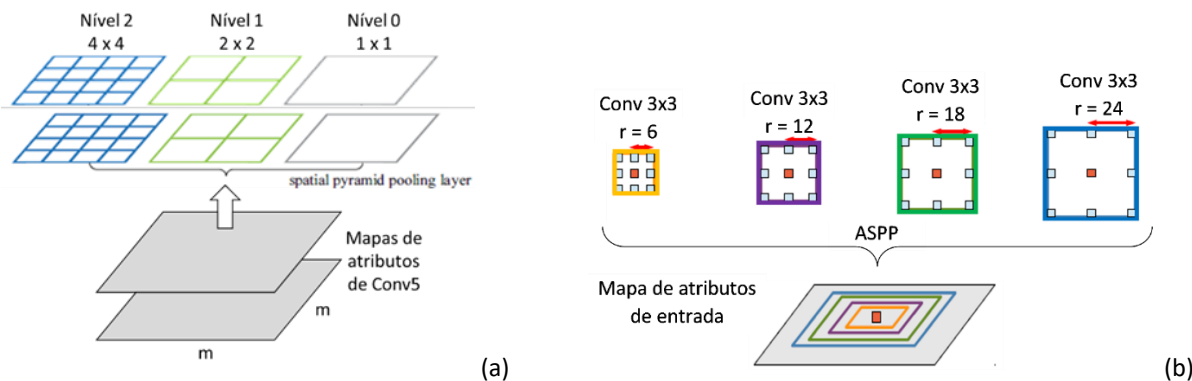
**Figura 2.93** – Arquitetura da rede DeepLab-LargeFoV com grande campo de visão.

Para treinamento das redes DeepLab, a rede VGG-16 pré-treinada no ImageNet é empregada e ajustada com *fine-tuning* na tarefa de classificação de *pixel* com 21 classes no conjunto PASCAL-VOC. O melhor desempenho foi obtido através da exploração de conexões de salto e grande campo de visão da rede DeepLab-MSc-LargeFOV.

### 2.4.8.1 Modelo DeepLab-v2

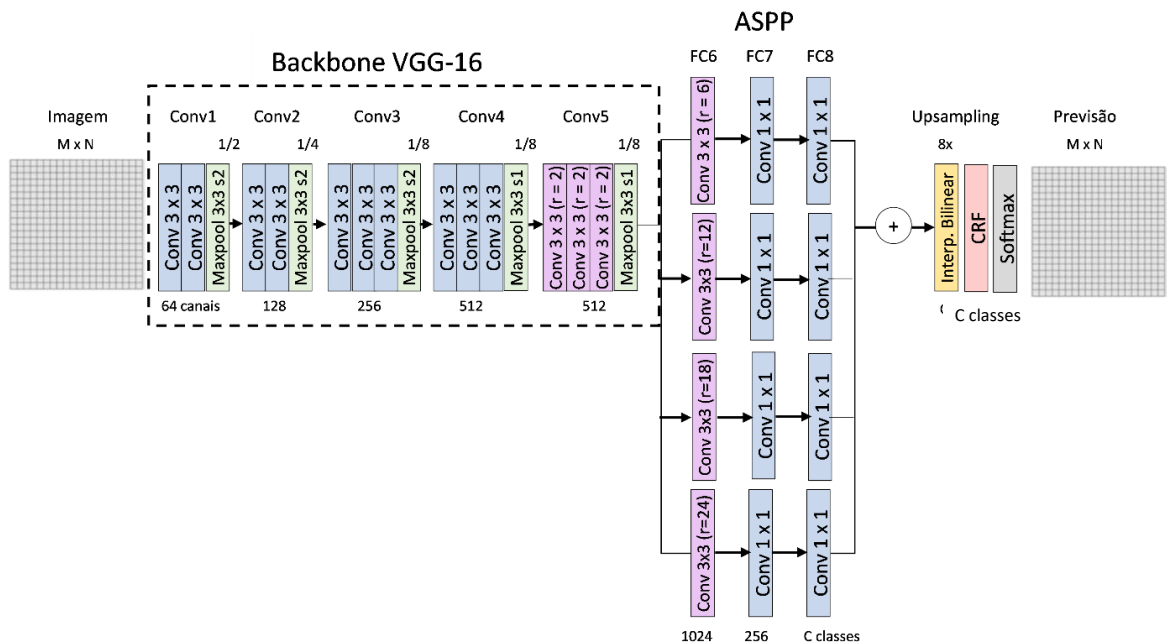
Na segunda versão da rede DeepLab [[CHEN et al. 2018a](#)], o módulo *Atrous Spatial Pyramid Pooling* (ASPP) foi proposto para segmentar objetos de forma robusta em várias escalas. O ASPP é inspirado no sucesso da pirâmide espacial (*Spatial Pyramid Pooling*) da [Figura 2.94a](#), usada no topo da última camada convolucional na rede de classificação de imagens SPPNet [[HE et al. 2015b](#)], que mostrou ser eficaz para reamostrar atributos em diferentes escalas antes da classificação nas camadas FC. O nível de pirâmide mais grosseiro tem um único compartimento de “*pooling global*” que cobre todo o mapa de atributos de Conv5. Em cada nível de pirâmide de  $n \times n$  bins, cada compartimento espacial agrupa a resposta de um filtro *maxpooling* deslizando não sobreposto no mapa de atributos de entrada.

Na [Figura 2.94b](#), para classificar o *pixel* central (vermelho), o ASPP extrai informações multiescala de uma camada de atributos de entrada usando vários filtros *atrous* paralelos com diferentes taxas de amostragem e campos de visão, capturando objetos e contexto em várias escalas. Os campos de visão são mostrados em cores diferentes no mapa de atributos de entrada.

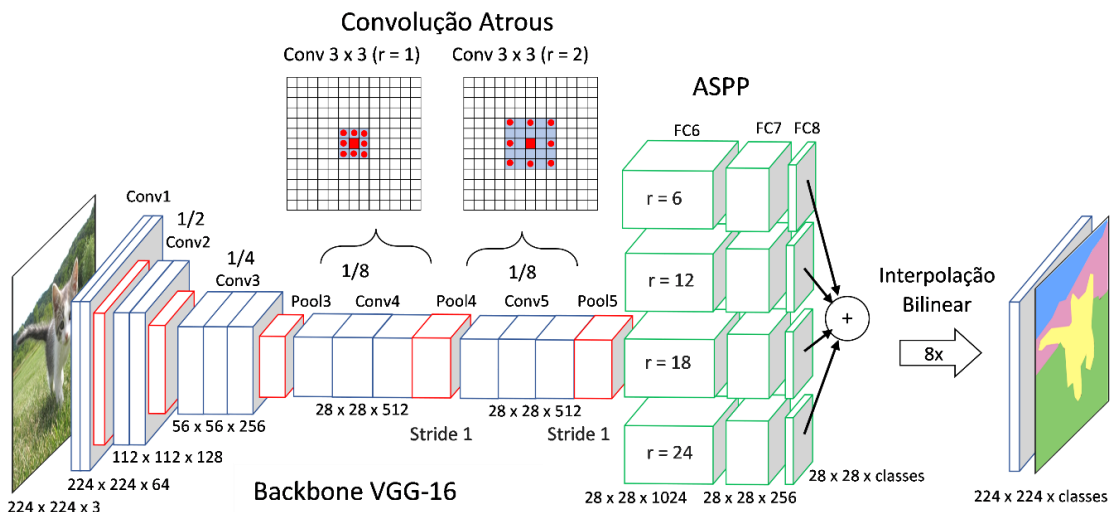


**Figura 2.94** – Pirâmide espacial de agrupamentos (SPP) [[HE et al. 2015b](#)] (a); Pirâmide espacial de agrupamentos atrous (ASPP) [[CHEN et al. 2018a](#)] (b).

Na rede DeepLab-v2, uma rede totalmente convolucional baseada na rede VGG-16 [SIMONYAN e ZISSERMAN 2015] ou ResNet-101 [HE *et al.* 2016a] (pré-treinada com ImageNet na tarefa de classificação de imagens) é adaptada e ajustada à tarefa de segmentação semântica. Conforme mostrado nas Figuras 2.95 e 2.96, o DeepLab-v2 com *backbone* VGG-16 usa um método ASPP com quatro ramos paralelos em FC6-FC7-FC8, com taxas  $r = \{2, 4, 8, 12\}$  (na variante ASPP-*Small*) ou  $r = \{6, 12, 18, 24\}$  (na variante ASPP-*Large*), se diferenciando do modelo DeepLab-LargeFOV de linha de base na Figura 2.93, que tem um único ramo com  $r = 12$ . Todas as convoluções em FC6 usam *kernels*  $3 \times 3$ , mas diferentes taxas de dilatação  $r$  para capturar objetos de tamanhos diferentes, sendo que o modelo ASPP-*Large* captura melhor o contexto da imagem em várias escalas. Os mapas de atributos multiescala de ASPP em FC6 passam por duas convoluções  $1 \times 1$  para diminuir o número de canais para  $C = 21$  e, em seguida, são agregados por adição antes das operações de *upsampling* de 8x, CRF e *softmax* para obter o mapa de previsão final.



**Figura 2.95** – Arquitetura da rede DeepLab-v2 com *backbone* VGG-16 e ASPP-*Large*. Baseado em [CHEN *et al.* 2018a].



**Figura 2.96** – Mapas de atributos de DeepLab-v2 com *backbone* VGG-16 e ASPP-*Large*.

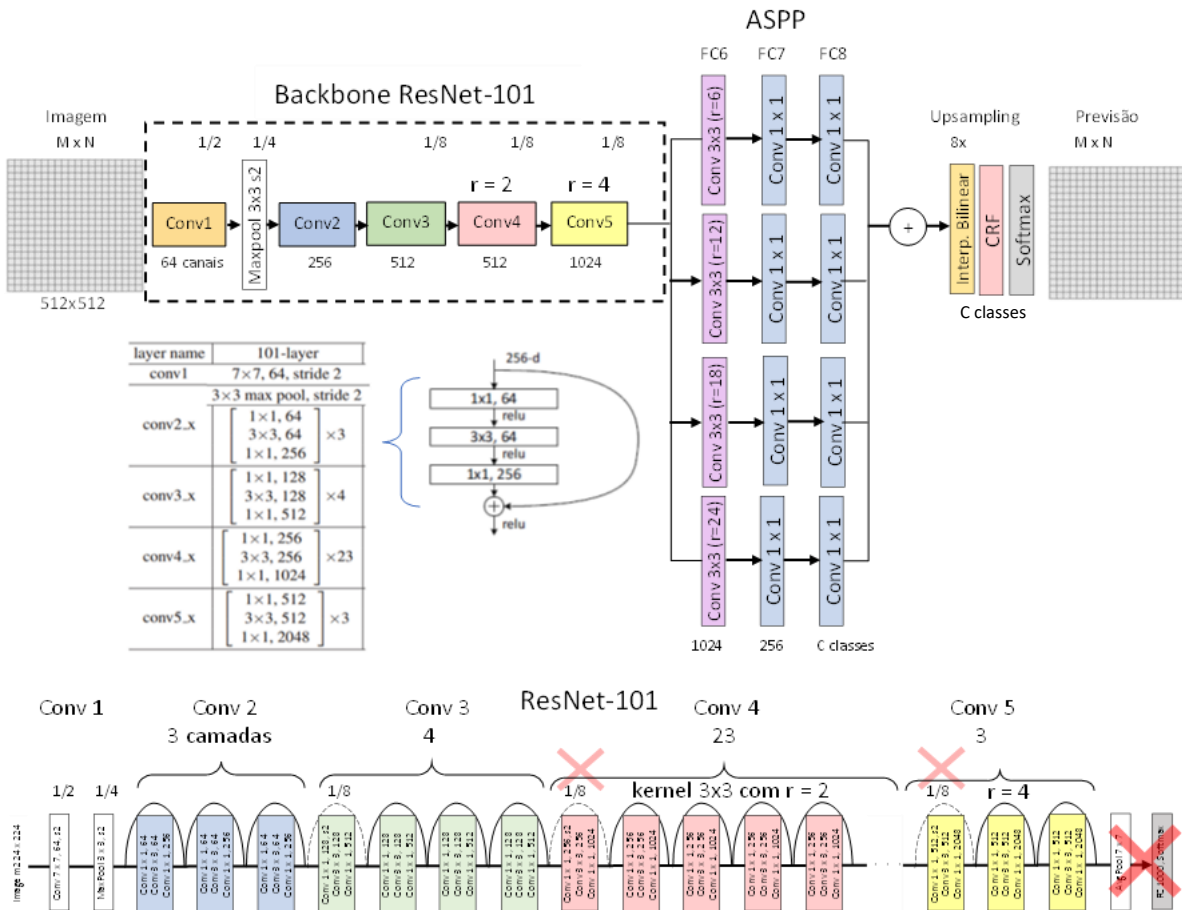


Figura 2.97 – Arquitetura da rede DeepLab-v2 com backbone ResNet-101 usando ASPP.

A Figura 2.97 mostra a versão DeepLab-v2 com a rede residual ResNet-101 [HE *et al.* 2016a] como *backbone*. Para diminuir o fator de redução da resolução espacial de entrada de 32 para 8, as operações de subamostragem dos blocos Conv4 e Conv5 de ResNet não são executadas. Desta forma, para obter um maior campo receptivo, a convolução  $3 \times 3$  dos blocos Conv4 e Conv5 deve ser substituída por convolução *atrous*  $3 \times 3$  com taxa de dilatação  $r = 2$  em Conv4 e  $r = 4$  em Conv5. Esta variante de rede residual obteve melhor desempenho de segmentação semântica em comparação com o modelo original DeepLab-v2 baseado em VGG-16.

Resumindo, na rede DeepLab-v2, uma rede neural convolucional de classificação – como VGG-16 ou ResNet-101 – é modificada e empregada de maneira totalmente convolucional, removendo algumas camadas de *maxpooling* (para reduzir o fator de saída de 32 para 8) e usando convolução *atrous* para aumentar o campo de visão. Em seguida, a pirâmide espacial ASPP captura atributos multiescala por meio de quatro ramos de convoluções *atrous* em paralelo. No final, um estágio de interpolação bilinear aumenta os mapas de atributos por um fator de 8 para recuperar a resolução da imagem original. Além disso, um CRF totalmente conectado é aplicado para refinar o resultado da segmentação e capturar melhor os limites do objeto. Por fim, o classificador de 1.000 vias do ImageNet na última camada é substituído por um classificador com o número de classes da tarefa específica, com  $C = 21$  classes para PASCAL-VOC. A camada *softmax* gera o mapa de previsão final.

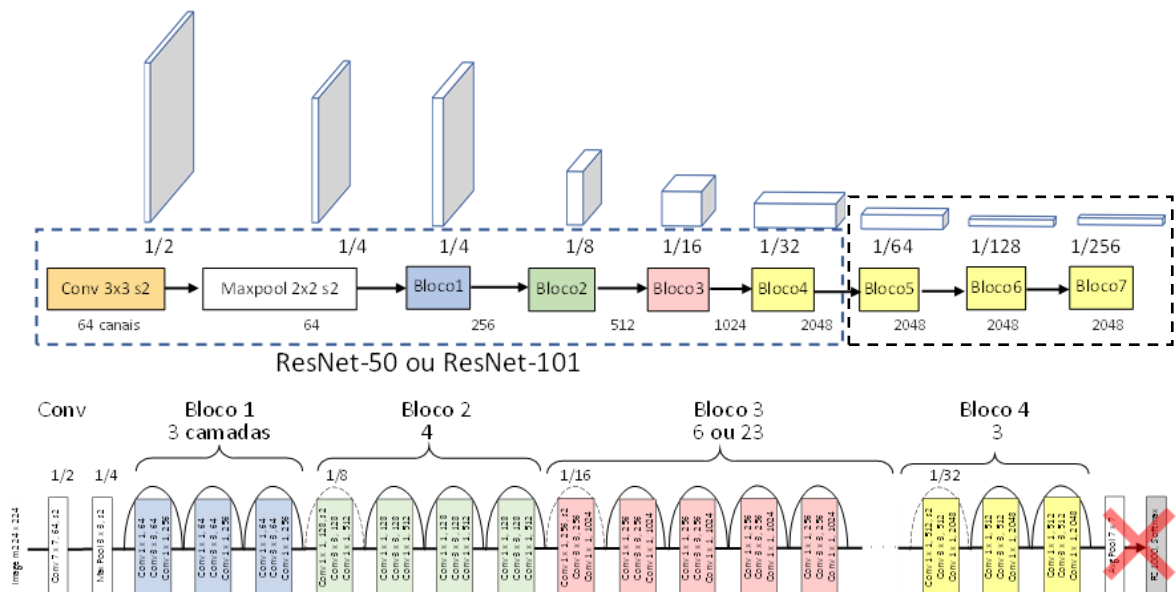
### 2.4.8.2 Modelo DeepLab-v3

Para analisar o problema de segmentação de objetos em múltiplas escalas, Chen *et al.* compararam duas estratégias em DeepLab-v3 [CHEN *et al.* 2017]: módulos de contexto que empregam convolução *atrous* em série ou em paralelo, com diferentes taxas de dilatação para capturar informação contextual multiescala. Ambas usam como *backbone* uma rede ResNet adaptada [HE *et al.* 2016a], pré-treinada no ImageNet, com remoção da subamostragem e aplicação da convolução *atrous* nas duas últimas camadas para preservar a resolução e extrair atributos densos. No modelo DeepLab-v3 em série, um módulo extra de convoluções *atrous* pode produzir grandes campos receptivos, como exemplificado na Figura 2.89a, mas não captura atributos multiescala. No modelo DeepLab-v3 em paralelo, que usa um módulo ASPP modificado (*Atrous Spatial Pyramid Pooling*) desenvolvido no DeepLab-v2 [CHEN *et al.* 2018a], várias camadas *atrous* têm a mesma entrada e suas saídas são concatenadas em vez de somadas como na versão anterior. Assim, o mapa de atributos de saída do ASPP obtém informações da entrada com diferentes escalas de campos receptivos, capturando diferentes tamanhos de contexto.

#### a) Modelo de convoluções *atrous* em série

A Figura 2.98 mostra uma rede hipotética baseada na ResNet [HE *et al.* 2016a], onde são realizadas a convolução padrão e subamostragem. O bloco 4 (amarelo) da ResNet foi copiado nos blocos 5, 6 e 7, organizados em série no módulo extra. Assim, os blocos 4 a 7 têm três convoluções residuais  $3 \times 3$  cada, além de uma subamostragem (arco tracejado) com convolução de passo 2, por bloco (exceto no bloco 7). Desta forma, a representação da imagem inteira  $M \times N$  é resumida no último mapa de atributos de menor resolução espacial, com fator de redução de 256 no bloco 7, conforme ilustrado na Figura 2.98.

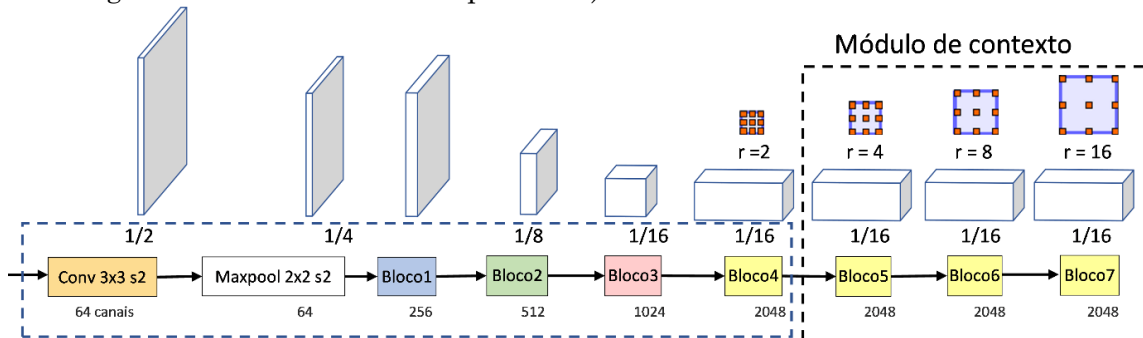
A subamostragem permite a captura de informações de maior contexto nos blocos mais profundos (Figura 2.55). No entanto, operações consecutivas de subamostragem (como *pooling* ou convolução com *stride* > 1 em CNNs profundas) levam a uma diminuição significativa na resolução inicial da imagem e são prejudiciais à segmentação semântica, porque a informação espacial é perdida nas camadas mais profundas.



**Figura 2.98** – Blocos em série sem convolução *atrous*, com fator de redução de 256. Adaptado de [CHEN *et al.* 2017].



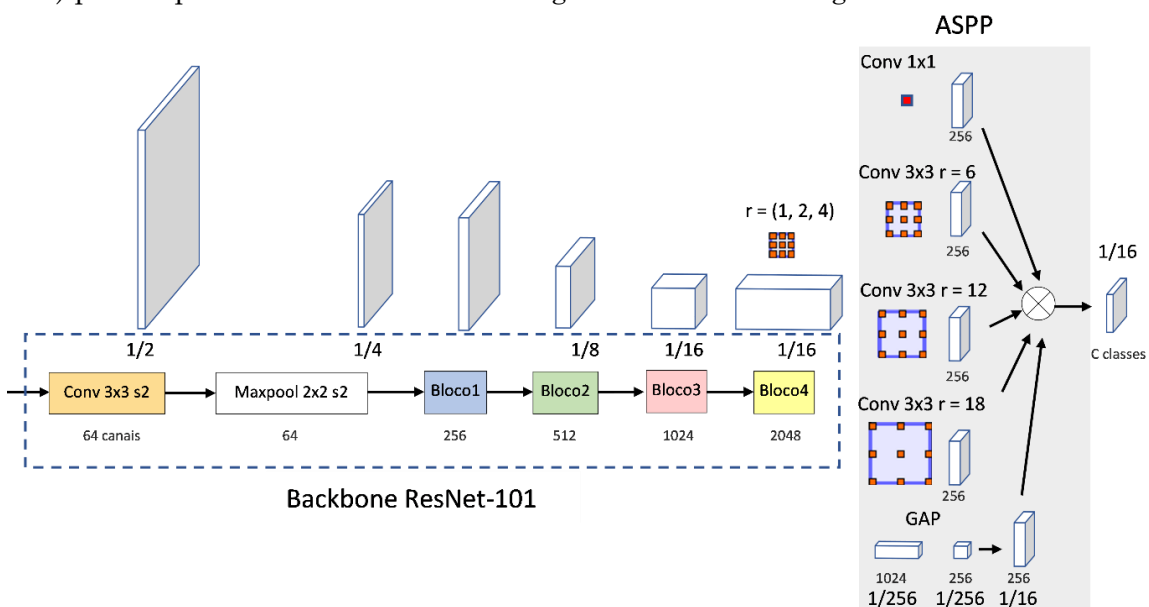
Para resolver este problema, foi proposto um modelo DeepLab-v3 com convolução *atrous* em série. Na **Figura 2.99**, a convolução *atrous* permite extrair mapas de atributos mais densos, removendo as operações de redução da amostragem (*pooling*) dos blocos 4 a 7, mantendo um fator de redução de 16. A convolução *atrous* permite manter a resolução e simultaneamente aumentar o campo de visão, sem aumentar o número de parâmetros (**Figura 2.89a**). Um método *multi-grid* também foi proposto com diferentes taxas *atrous* ( $r_1, r_2, r_3$ ) para as três camadas convolucionais dentro de cada bloco 4 a 7. Ao contrário da rede DilatedNet, o módulo de contexto com os blocos 5 a 7 em série é aplicado nos mapas de atributos de saída do bloco 4, em vez do mapa de previsão da CNN (que contém  $C$  canais de saída igual ao número de classes previstas).



**Figura 2.99** – DeepLab-v3 com convolução *atrous* em série, com fator de redução de 16. Adaptado de [CHEN *et al.* 2017].

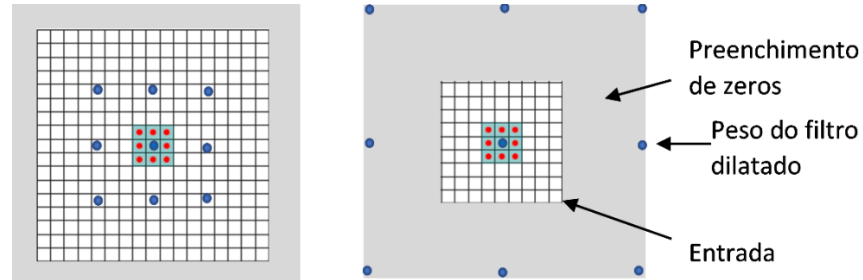
**b) Modelo de convoluções *atrous* em paralelo**

No modelo DeepLab-v3 [CHEN *et al.* 2017] com convoluções *atrous* em paralelo, o módulo ASPP original, proposto em DeepLab-v2 [CHEN *et al.* 2018a], onde quatro convoluções *atrous* paralelas com diferentes taxas  $r = (6, 12, 18, 24)$  são aplicadas no topo da CNN, foi modificado para capturar informações em várias escalas. Na **Figura 2.100**, o ASPP aprimorado consiste em: uma convolução  $1 \times 1$ , três convoluções *atrous*  $3 \times 3$  com taxas de dilatação  $r = (6, 12, 18)$ , concatenados com um agrupamento de média global (GAP) para capturar atributo de contexto global a nível de imagem.



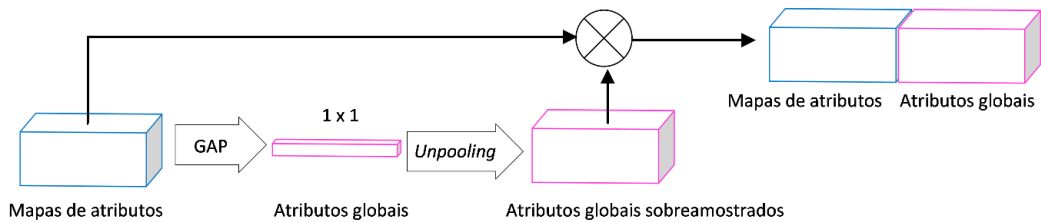
**Figura 2.100** – DeepLab-v3 com convolução *atrous* (ASPP) em paralelo com concatenação de atributos de contexto global de nível de imagem. Adaptado de [CHEN *et al.* 2017].

Um problema que ocorre ao aplicar uma convolução *atrous*  $3 \times 3$  com uma taxa grande ou, no caso extremo, com *kernel* dilatado maior que o tamanho do mapa de atributos de entrada, é que o número de pesos válidos do filtro (ou seja, os pesos que são aplicados à região de atributos válida, em vez de preenchidos com zeros) se torna menor. Com isso, em vez do filtro  $3 \times 3$  capturar informações de contexto global da imagem, ele acaba degenerando para um filtro  $1 \times 1$ , pois apenas o peso central do filtro é efetivo, como mostra a [Figura 2.101](#) [CHEN *et al.* 2017].

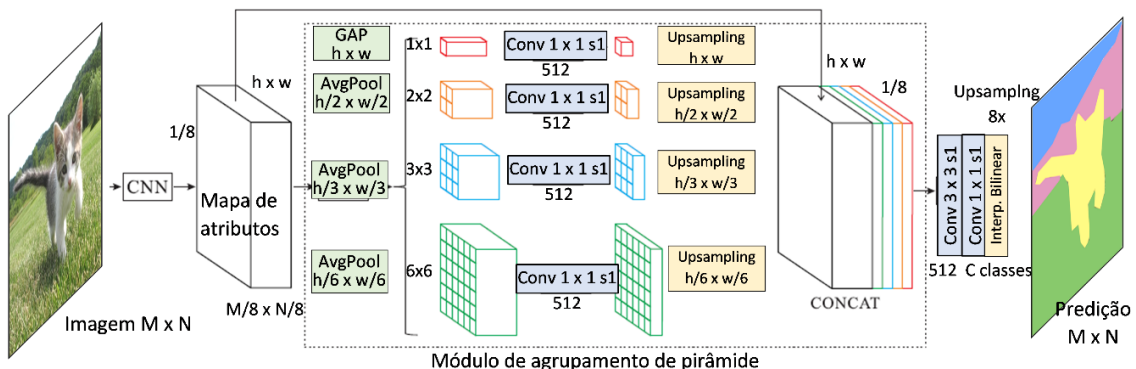


**Figura 2.101** – *Convolução atrous com taxa de dilatação muito grande: com kernel dilatado maior que entrada, apenas peso central é convoluído de forma semelhante a um kernel padrão  $1 \times 1$ .*

Para superar esse problema, foi proposto incorporar um atributo de nível de imagem no ASPP para adicionar informações de contexto global, influenciado pelo uso de *Global Average Pooling* (GAP) em ParseNet [LIU *et al.* 2015] e pela pirâmide espacial de *pooling* em PSPNet [ZHAO *et al.* 2017]. O atributo global de nível de imagem foi explorado no ParseNet para obter informações de contexto global, como mostra a [Figura 2.102](#). Assim, o GAP calcula o valor médio na área total  $h \times w$  do mapa de atributos, para cada canal da entrada, reduzindo a dimensão espacial para  $1 \times 1$ . A rede de segmentação de imagens PSPNet (*Pyramid Scene Parsing Net*) emprega um módulo de pirâmide espacial de *pooling* (*average pooling*) em quatro escalas de grade no topo da CNN-base, incluindo o agrupamento em nível global de imagem, para agregar informações de vários tamanhos de campos receptivos ([Figura 2.103](#)).



**Figura 2.102** – *Módulo de contexto de ParseNet com GAP. Adaptado de [LIU *et al.* 2015].*



**Figura 2.103** – *Módulo de contexto de PSPNet. Adaptado de [ZHAO *et al.* 2017].*



Particularmente, no DeepLab-v3 paralelo com *backbone* ResNet-101, mostrado na Figura 2.104, um agrupamento de média global (GAP) é incluído no módulo ASPP. Após o GAP, o atributo de nível de imagem é alimentado em uma convolução  $1 \times 1$  para diminuir o número de canais para 256 e, em seguida, uma interpolação bilinear do atributo aumenta a resolução espacial em 16x para concatenação com os outros mapas de atributos do ASPP.

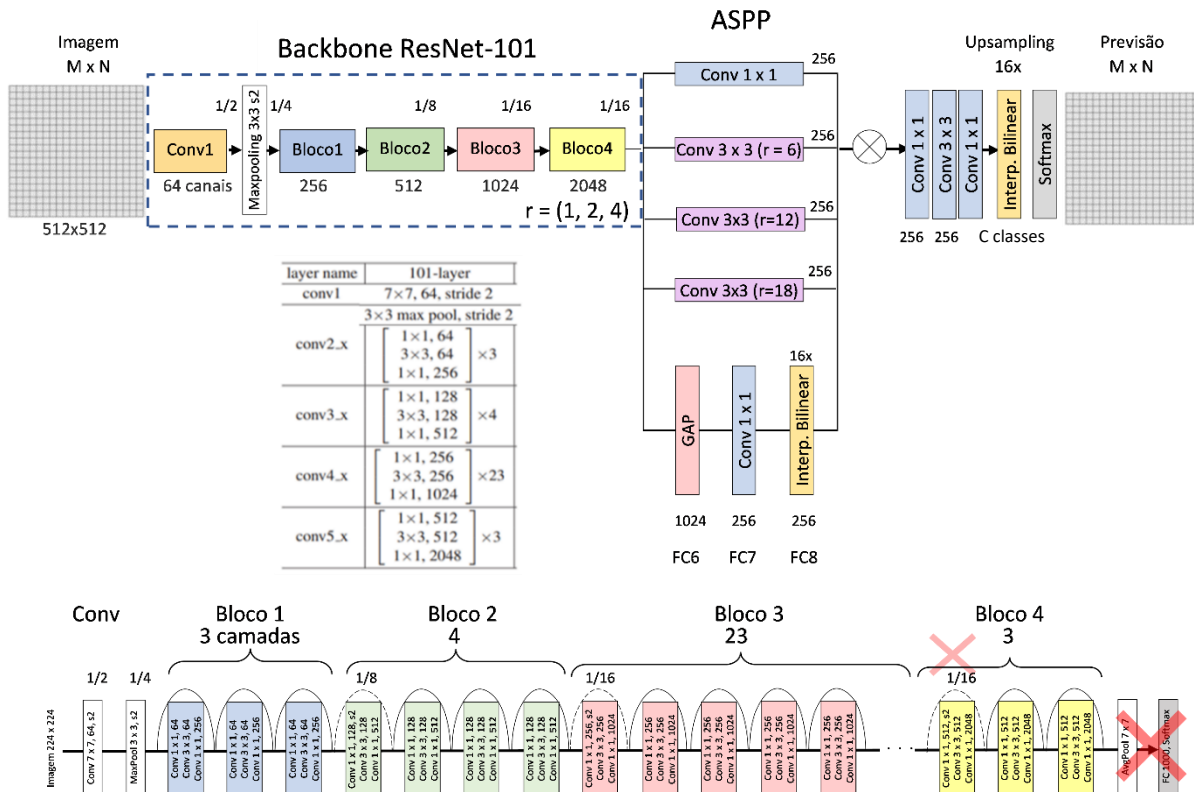


Figura 2.104 – DeepLab-v3 com backbone ResNet-101 usando ASPP modificado. Baseado em [CHEN et al. 2017].

Na Figura 2.104, os mapas de atributos resultantes de todas as ramificações do ASPP são concatenados (em vez de somados como em DeepLab-v2) e passam por outra convolução  $1 \times 1$  e  $3 \times 3$  (também com 256 filtros) antes da convolução  $1 \times 1$  final que gera as previsões finais com  $C$  mapas de atributos. Como a saída tem um fator de redução de 16 (diferente de DeepLab-v2 na Figura 2.97 com fator de 8), uma interpolação bilinear de 16x recupera a resolução original, antes da camada *softmax*.

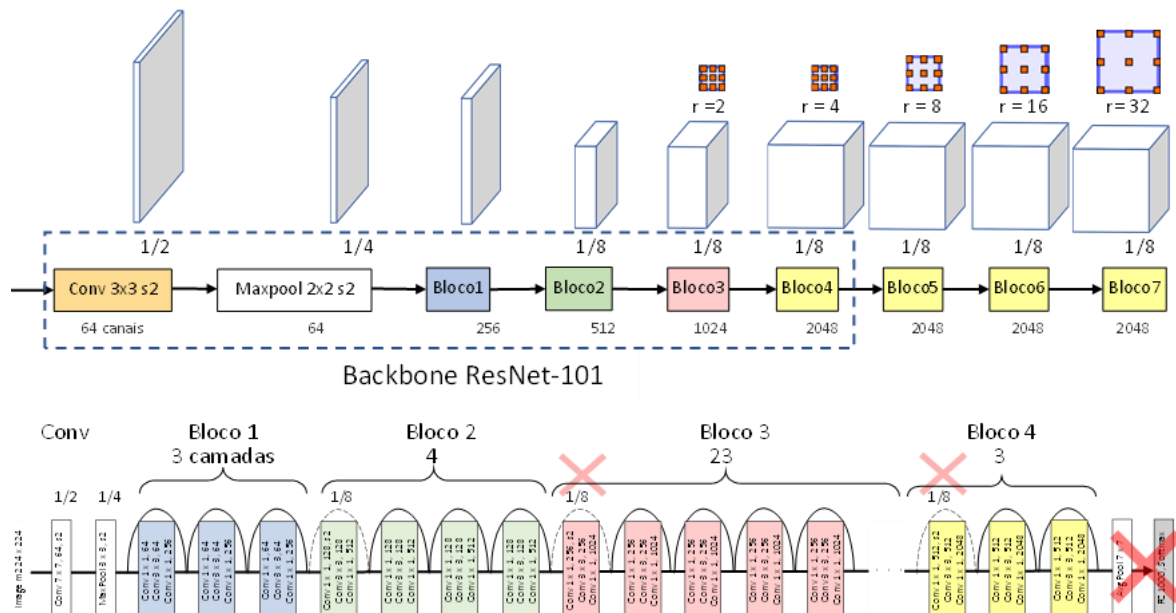
Seguindo o protocolo de treinamento das redes DeepLab-v1 e v2, blocos menores são cortados a partir da imagem original. Para que a convolução *atrous* com taxas maiores seja efetiva, é necessário um tamanho de bloco que gere mapas de atributos de tamanho suficiente na entrada do ASPP; caso contrário, os pesos dos filtros *atrous* no módulo ASPP serão aplicados à região de preenchimento zero, degenerando para uma convolução  $1 \times 1$  e, portanto, os filtros do ASPP não terão diferentes tamanhos de campos receptivos para obter informações de contexto em diferentes escalas. Assim, um tamanho de corte de 512 *pixels* foi empregado para gerar os blocos de imagens de treinamento da rede DeepLab-v3, gerando mapas de atributos de tamanho  $32 \times 32$  na entrada do módulo ASPP, quase do tamanho do campo de visão do maior filtro expandido com taxa de dilatação  $r = 18$ .

A rede DeepLab-v3 não faz pós-processamento com CRF, mas ainda tem um melhor desempenho em relação às versões anteriores no conjunto de teste PASCAL-VOC 2012.

### 2.4.8.3 Modelo DeepLab-v3+

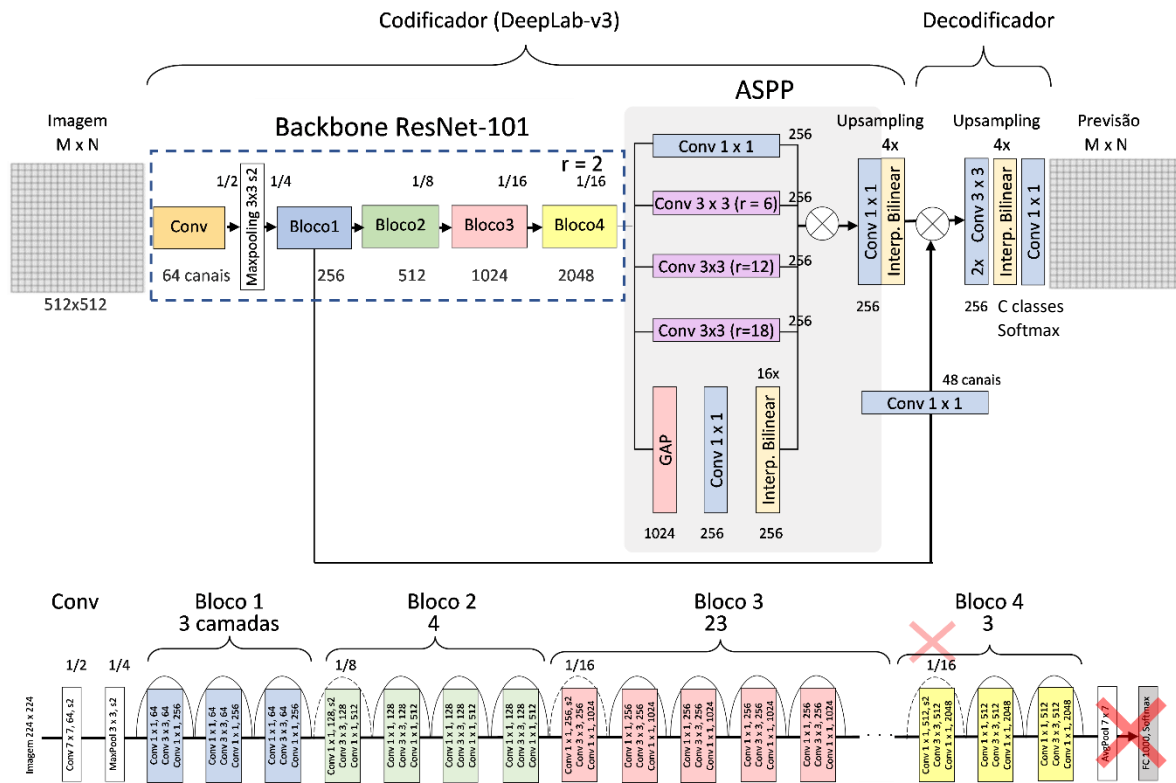
Para capturar informação contextual em múltiplas escalas, o DeepLab-v3 [CHEN *et al.* 2017] possui um módulo ASPP com convoluções *atrous* paralelas com diferentes taxas de dilatação e contexto global. No entanto, embora mais informações semânticas sejam codificadas no último mapa de atributos, informações detalhadas relacionadas aos limites do objeto estão ausentes devido às operações de subamostragem nas camadas do *backbone*, com redução da resolução da entrada por um fator de 16.

Este problema poderia ser amenizado, eliminando a operação de subamostragem nas camadas intermediárias do *backbone* para preservar a resolução com um fator de 8 ou 4, e também aplicando convoluções *atrous* para extrair mapas de atributos mais densos. No entanto, devido à memória limitada das GPUs, não é computacionalmente viável extrair mapas de atributos tão grandes com redes neurais muito profundas, tais como ResNet [HE *et al.* 2016a] e Xception [CHOLLET 2017]. Por exemplo, removendo a subamostragem dos blocos 3 e 4 do *backbone* ResNet-101, 23 camadas convolucionais residuais do bloco 3 mais três camadas residuais do bloco 4 seriam impactadas pela computação intensiva da convolução *atrous* em mapas de entrada maiores, como mostra a rede hipotética da Figura 2.105.



**Figura 2.105** – Módulos em série com convolução *atrous*, com fator de redução de 8. Baseado em [CHEN *et al.* 2018b].

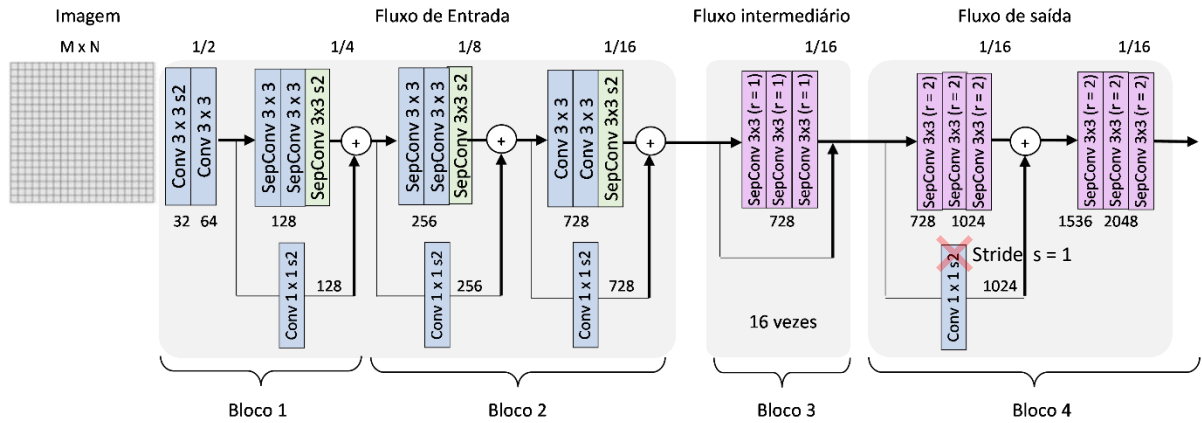
Por outro lado, como a resolução do mapa de atributos de saída do DeepLab-v3 mostrado na Figura 2.104 é recuperada com uma interpolação bilinear com fator de 16, isso resulta em perda de detalhes de segmentação de objetos. Assim, para recuperar gradualmente os detalhes dos limites do objeto na rede DeepLab-v3+, sem aumentar a resolução da saída da rede, uma solução viável foi estender o módulo codificador do DeepLab-v3 [CHEN *et al.* 2017], adicionando um módulo decodificador com conexões de salto. Portanto, a rede DeepLab-v3+ [CHEN *et al.* 2018b] mostrada na Figura 2.106 combina o módulo ASPP no codificador, capaz de codificar informações contextuais em várias escalas, com a arquitetura codificador-decodificador com conexões de salto para refinar os resultados da segmentação.



**Figura 2.106** – DeepLab-v3+ com backbone ResNet-101 usando ASPP e decodificador. Baseado em [CHEN et al. 2018b].

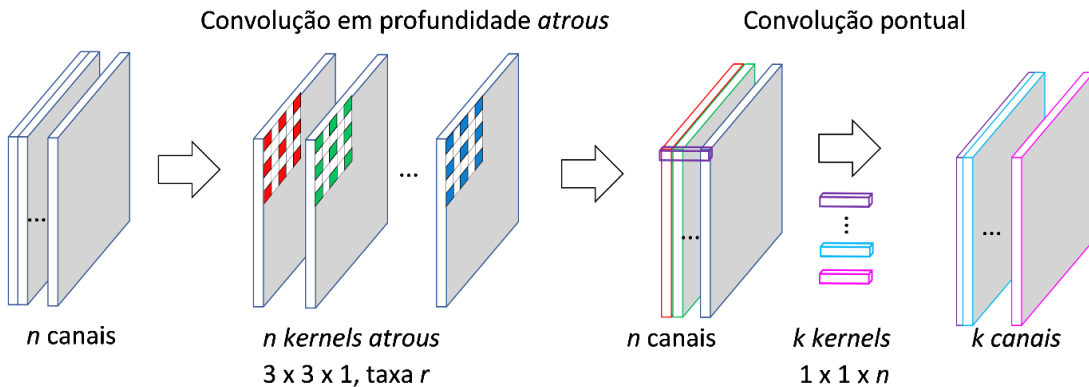
Na Figura 2.106, o DeepLab-v3+ usa a rede DeepLab-v3 com *backbone* ResNet-101 como codificador, ou seja, com fator de saída 16. No decodificador, os mapas de atributos de saída do codificador após o ASPP são sobreamostrados bilinearmente por um fator de 4 e, em seguida, concatenados com os mapas de atributos da camada rasa do *backbone*, que têm a mesma resolução espacial com fator 4 (saída do bloco 1). Uma convolução  $1 \times 1$  é aplicada aos atributos de baixo nível da camada rasa para reduzir o número de canais para 48, pois as camadas rasas geralmente contêm muitos canais que dificultam o treinamento. Após a concatenação da saída do codificador com o mapa de atributos de baixo nível, duas convoluções  $3 \times 3$  com 256 canais são aplicadas para refinar os atributos e obter resultados de segmentação mais nítidos. A saída é seguida de outra interpolação bilinear de 4x para recuperar a resolução da entrada.

Seguindo a evolução das redes de classificação, a rede Xception original [CHOLLET 2017] na Figura 2.44 foi adaptada para ser usada como *backbone* da rede de segmentação semântica DeepLab-v3+. No Xception modificado para DeepLab, mostrado na Figura 2.107, chamado Xception-DeepLab, todas as operações de *maxpooling* do fluxo de entrada da rede Xception original foram substituídas por convolução separável em profundidade (caixas verdes), usando passo 2 nos blocos 1 e 2 para subamostragem. O número de repetições é aumentado de 8 para 16 no fluxo intermediário. Além disso, a rede Xception modificada aplica uma convolução separável em profundidade *atrous* (caixas rosa) com taxas  $r = 1$  e 2 nos blocos 3 (fluxo intermediário) e 4 (fluxo de saída), respectivamente, para reduzir a complexidade na extração de mapas de atributos densos e aumentar o campo receptivo, mantendo a resolução em 1/16 da entrada original.



**Figura 2.107** – Módulo codificador *Xception* modificado (*Xception-DeepLab*) com convolução separável em profundidade *atrous*. Baseado em [CHEN et al. 2018b].

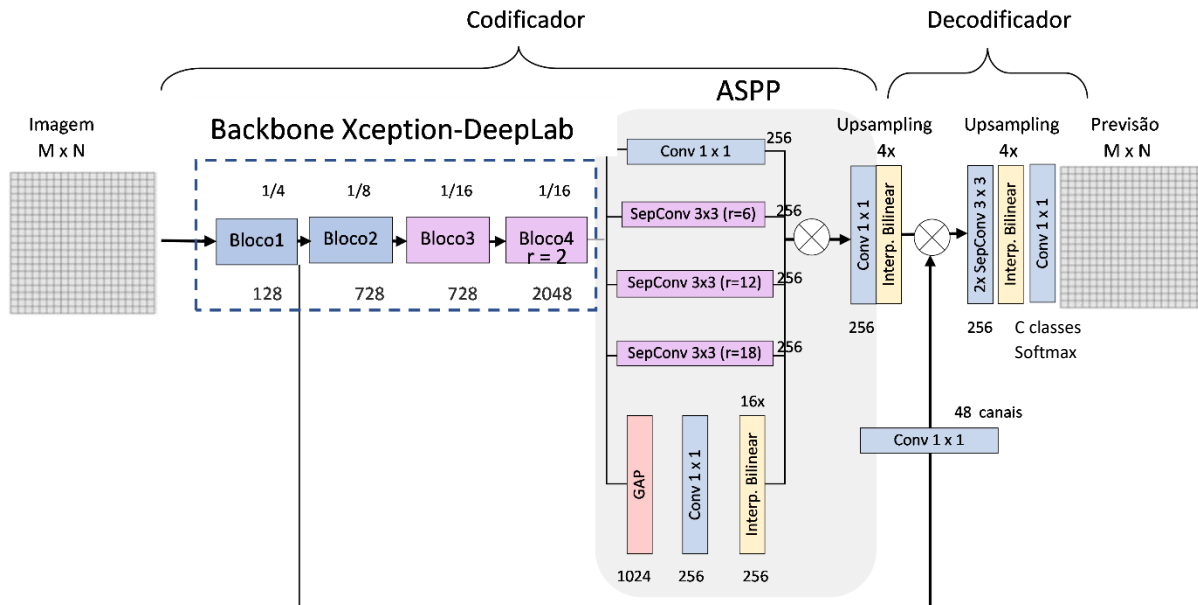
Na convolução separável em profundidade *atrous*, a convolução *atrous* é adotada na convolução em profundidade (ou seja, convolução espacial  $3 \times 3$ ) com taxa de dilatação  $r$ , seguida por uma convolução pontual  $1 \times 1$ , conforme ilustrado na Figura 2.108. A convolução em profundidade realiza uma convolução espacial *atrous* independentemente para cada canal de entrada, enquanto a convolução pontual combina a saída da convolução em profundidade.



**Figura 2.108** – Convolução separável em profundidade *atrous*. Adaptado de [CHEN et al. 2018b].

A Figura 2.109 mostra os detalhes da arquitetura da rede DeepLab-v3+ que usa uma estrutura codificadora com *backbone* *Xception-DeepLab* e um módulo ASPP no final do codificador, sendo que ambos usam a convolução separável em profundidade *atrous* (caixas rosa). O decodificador usa duas convoluções separável em profundidade padrão para refinar os mapas de atributos da saída do codificador com fator 4, antes de passar por uma interpolação bilinear de 4x para recuperar a resolução da entrada, uma convolução  $1 \times 1$  com  $C$  mapas de atributos e uma camada *softmax* para previsão final.

Seguindo a mesma estrutura da rede DeepLab-v3+ na Figura 2.109, a rede MRFN (*Multi Receptive Field Network*) [YUAN et al. 2020] usa um caminho paralelo de convoluções dilatadas com  $r = 2$  nos blocos do fluxo intermediário e  $r = 4$  nos blocos do fluxo de saída da rede *Xception-DeepLab* na Figura 2.107, fundindo os atributos com diferentes campos receptivos na saída de cada bloco para capturar contexto multiescala no *backbone*.



**Figura 2.109** – Arquitetura da rede *DeepLab-v3+* com backbone *Xception-DeepLab*. Baseado em [CHEN *et al.* 2018b].

## 2.4.9 Modelo DenseASPP

Em aplicações de direção autônoma existe uma grande mudança de escala de objetos (carros, pessoas, etc.) causada pela distância da câmera, o que representa um desafio na codificação das informações multiescalas para representação de atributos de alto nível. A convolução *atrous* ameniza este problema permitindo gerar mapas de atributos com campos receptivos maiores sem sacrificar a resolução espacial. Porém, apesar de obter vários tamanhos de campos receptivos, um mapa semântico gerado com convoluções *atrous* consecutivas, como DilatedNet e DeepLab-v3 em série, ainda sofre de limitações para capturar atributos multiescala. Em outras palavras, todos os neurônios no mapa de atributos de saída compartilham o mesmo tamanho fixo de campo receptivo, representando apenas atributos em escala única [YANG *et al.* 2018].

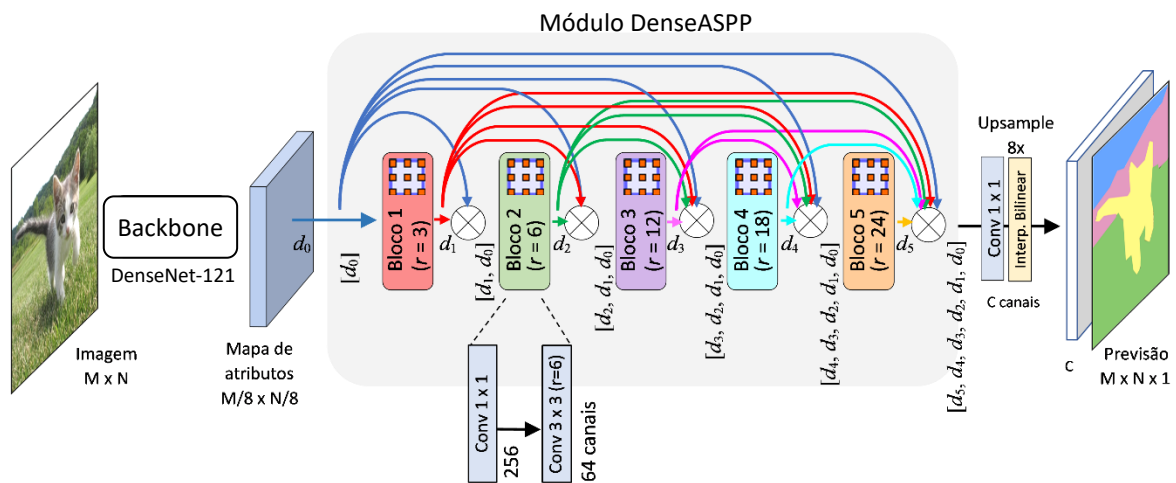
Tanto a rede PSPNet, que usa um módulo de pirâmide de *pooling*, como as redes DeepLab-v2, DeepLab-v3 e DeepLab-v3+, que usam um módulo ASPP, concatenam atributos de vários tamanhos de campos receptivos para previsão final, mostrando que informações multiescalas resultam em uma segmentação mais robusta. Especificamente, a rede PSPNet emprega uma pirâmide espacial com quatro camadas de *pooling* em paralelo para agregar informações de vários tamanhos de campos receptivos. Já o módulo ASPP concatena mapas de atributos gerados pelas convoluções *atrous* em paralelo, com diferentes taxas de dilatação, de modo que os neurônios no mapa de saída incorporem atributos em vários tamanhos de campos receptivos. Porém, embora o ASPP seja capaz de codificar atributos multiescala, a resolução do mapa no eixo da escala não é densa o suficiente para algumas aplicações [YANG *et al.* 2018].

Além disso, o ASPP ainda sofre de outra limitação. Quando as imagens são de alta resolução, os neurônios devem ter um campo receptivo ainda maior e uma taxa de dilatação grande deve ser empregada. No entanto, conforme a taxa de dilatação aumenta (por exemplo,  $r > 24$ ), a convolução *atrous* com filtro muito esparso torna-se cada vez mais



ineficaz, ocorrendo deterioração do *kernel*. Portanto, uma estrutura de rede capaz de simultaneamente codificar informações em várias escalas de forma densa e atingir um tamanho suficientemente grande de campo receptivo foi desenvolvida na rede DenseASPP (*Densely Connected Atrous Spatial Pyramid Pooling*) [YANG *et al.* 2018] para resolver este problema.

A Figura 2.110 ilustra a estrutura da rede DenseASPP, que consiste em uma rede-base modificada, por exemplo, *backbone* DenseNet-121 [HUANG *et al.* 2017] com fator de saída 8 (sem subamostragem nos dois últimos blocos de transição), seguido por um módulo DenseASPP com cinco blocos de convolução *atrous* em série, com *kernel*  $3 \times 3$  com taxa de dilatação  $r \leq 24$ , onde a taxa de dilatação aumenta a cada bloco (por exemplo,  $r = 3, 6, 12, 18$  e  $24$ ). Devido à pilha de blocos de convoluções *atrous* consecutivas, os neurônios nos blocos posteriores obtêm um campo receptivo cada vez maior (Figura 2.89a), sem sofrer o problema de degradação do *kernel*.



**Figura 2.110** – Arquitetura DenseASPP com *backbone* DenseNet-121 modificado e módulo DenseASPP, onde a saída  $d_l$  de cada bloco convolucional *atrous*  $l$  é concatenada com os atributos dos blocos anteriores, gerando um mapa de atributos  $[d_{l-1}, d_{l-2}, \dots, d_0]$  que alimenta o próximo bloco. Adaptado de [YANG *et al.* 2018].

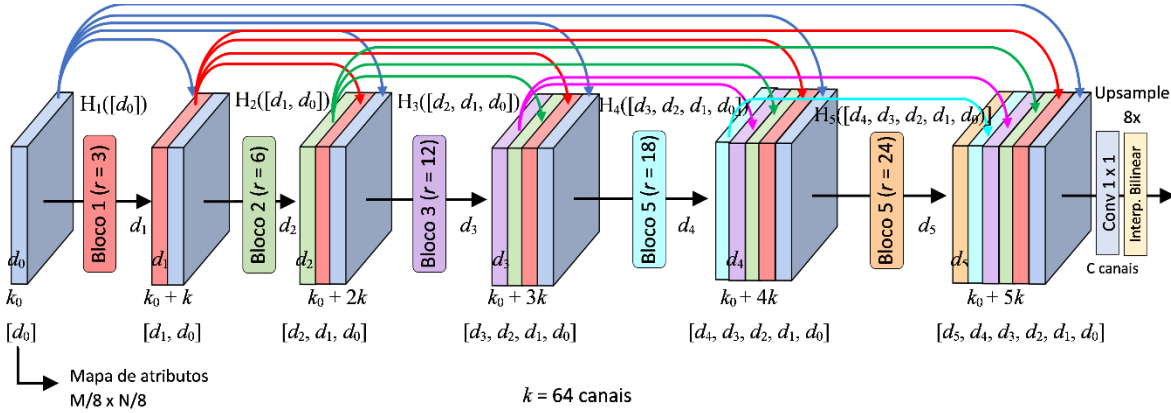
O mapa de atributos  $d_0$  de entrada do módulo DenseASPP é a saída do bloco 5 do *backbone*, com fator de redução de resolução espacial de 8. Cada bloco *atrous* no DenseASPP é representado pela Equação 2.6:

$$d_l = H_{k,r_l}([d_{l-1}, d_{l-2}, \dots, d_0]) \quad (2.6)$$

onde  $d_l$  é a saída do bloco  $l$ ,  $k$  é o tamanho do *kernel*,  $r_l$  representa a taxa de dilatação do bloco  $l$ , e  $[\cdot \cdot \cdot]$  denota a operação de concatenação, ou seja,  $[d_{l-1}, d_{l-2}, \dots, d_0]$  significa o mapa de atributos formado pela concatenação da entrada  $d_0$  com as saídas de todos os blocos anteriores ao bloco  $l$  [YANG *et al.* 2018].

De forma parecida ao módulo DenseNet da CNN de classificação DenseNet [HUANG *et al.* 2017], o módulo DenseASPP conecta um conjunto de blocos convolucionais *atrous* com conexões densas, onde a saída  $d_l$  de cada bloco  $l$  é concatenada a todas as saídas dos blocos anteriores, e passada ao bloco adiante, conforme ilustrado na Figura 2.111. Para controlar o tamanho do modelo, uma camada convolucional  $1 \times 1$  é adicionada ao bloco *atrous*, antes de cada camada de convolução *atrous*, para reduzir a profundidade do mapa de atributos. Através das concatenações de atributos, os neurônios do mapa de atributos

$[d_{l-1}, \dots, d_0]$  na saída de cada bloco  $l$  codificam informações semânticas de várias escalas e de forma densa. Assim, a saída final do DenseASPP é um mapa de atributos gerado por convoluções *atrous* de múltiplas taxas e multiescala (diferentes tamanhos de campos receptivos).



**Figura 2.111** – Mapas de atributos da rede DenseASPP com multiescala densa.

Para simplificar as notações seguintes, uma camada de convolução *atrous* simples é denominada  $H_{k,r}(x)$ , onde  $k$  é o tamanho do filtro,  $r$  é a taxa de dilatação e  $x$  é a entrada. A taxa de dilatação aumenta o campo receptivo de um *kernel* de convolução. Para uma camada convolucional *atrous* com taxa de dilatação  $r$  e tamanho de *kernel*  $k$ , o tamanho do campo receptivo é:

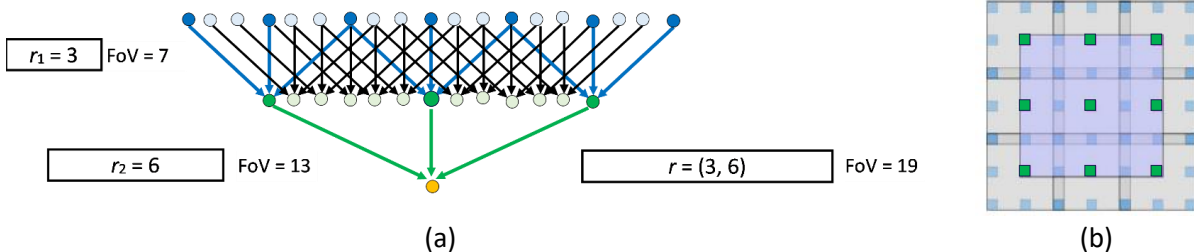
$$FoV_{k,r} = (r - 1) \cdot (k - 1) + k, \tag{2.7}$$

onde  $r - 1$  é o número de zeros inseridos entre os elementos do *kernel* dilatado e  $k$  é o número de elementos ou tamanho do *kernel*. Por exemplo, uma camada convolucional  $3 \times 3$  com  $r = 3$  tem um tamanho de campo receptivo igual a 7 [YANG *et al.* 2018].

No modo de convoluções *atrous* em série, usado nas redes DilatedNet e DeepLab-v3 (versão em série), camadas de convolução *atrous* com taxas de dilatação crescentes são empilhadas em cascata. Esta pilha de camadas convolucionais resulta em um campo receptivo ainda maior que cobre grandes objetos. Por exemplo, uma pilha de duas camadas de convolução *atrous*, com *kernel*  $k$  e taxas de dilatação  $r_1$  e  $r_2$ , tem um campo receptivo na saída dado por:

$$FoV = FoV_{k,r_1} + FoV_{k,r_2} - 1. \tag{2.8}$$

Deste modo, na Figura 2.112, uma pilha de camadas convolucionais com *kernels* de tamanho 3 e taxas de dilatação (3, 6) resulta em um campo receptivo de tamanho 19.



**Figura 2.112** – Campos receptivos ( $FoV = 7$  e  $13$ ) de convoluções *atrous* com taxas  $r_1 = 3$  e  $r_2 = 6$ .  $FoV = 19$  de duas convoluções *atrous* em série com taxas de dilatação  $r = (3, 6)$  (a); Versão 2D da convolução em série (b). Adaptado de [YANG *et al.* 2018].



No modo de convoluções *atrous* em paralelo, várias camadas *atrous* têm a mesma entrada e produzem mapas de atributos com diferentes escalas de campos receptivos concatenados na saída. Seguindo as Equações 2.7 e 2.8, o maior campo receptivo de ASPP(3, 6, 12, 18, 24) é o maior campo receptivo das convoluções *atrous* paralelas:

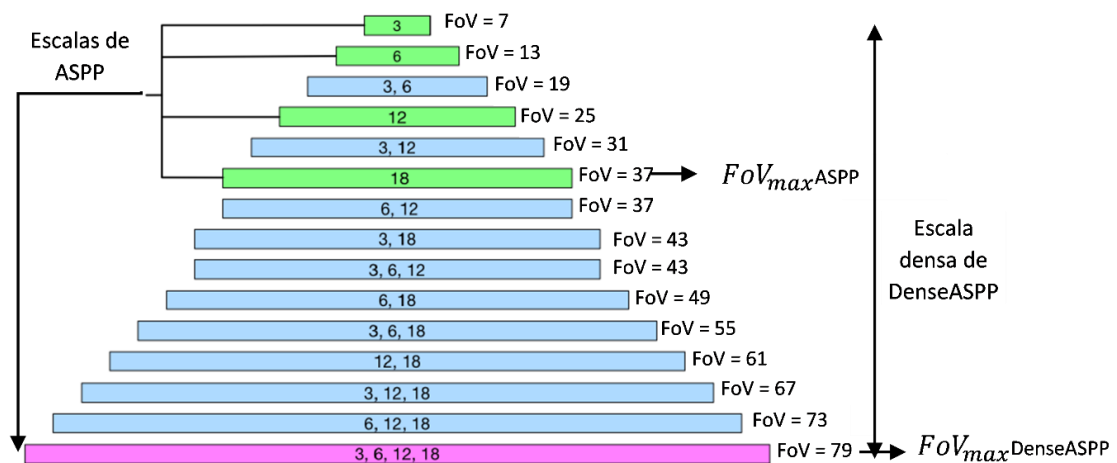
$$\begin{aligned} FoV_{maxASPP} &= \max[FoV_{3,3}, FoV_{3,6}, FoV_{3,12}, FoV_{3,18}, FoV_{3,24}] \\ &= \max[7, 13, 25, 37, 49] = 49. \end{aligned}$$

No entanto, o maior campo receptivo de DenseASPP(3, 6, 12, 18, 24) é a soma dos campos receptivos das convoluções *atrous* em série:

$$\begin{aligned} FoV_{maxDenseASPP} &= FoV_{3,3} + FoV_{3,6} + FoV_{3,12} + FoV_{3,18} + FoV_{3,24} - 4 \\ &= 7 + 13 + 25 + 37 + 49 - 4 = 127. \end{aligned}$$

Um campo receptivo máximo tão grande obtido pela rede DenseASPP(3, 6, 12, 18, 24) na Figura 2.110 fornece um contexto maior para objetos em imagens de alta resolução.

Comparado ao ASPP original, o módulo DenseASPP empilha todas as camadas dilatadas e as conecta por meio de conexões de salto densas. Essa mudança traz dois benefícios: pirâmide de atributos de escala mais densa com maior diversidade de escala, ou seja, maior resolução no eixo de escala; e campo receptivo maior, sem degradação do *kernel*. A Figura 2.113 mostra uma pirâmide de atributos simplificada do DenseASPP para ilustrar a diversidade de escala. Este DenseASPP é construído com taxa de dilatação de (3, 6, 12, 18). O(s) número(s) dentro de cada faixa representa(m) a combinação de diferentes taxas de dilatação em série, e o comprimento de cada faixa representa o campo receptivo do *kernel* de cada combinação em série (FoV) [YANG et al. 2018].



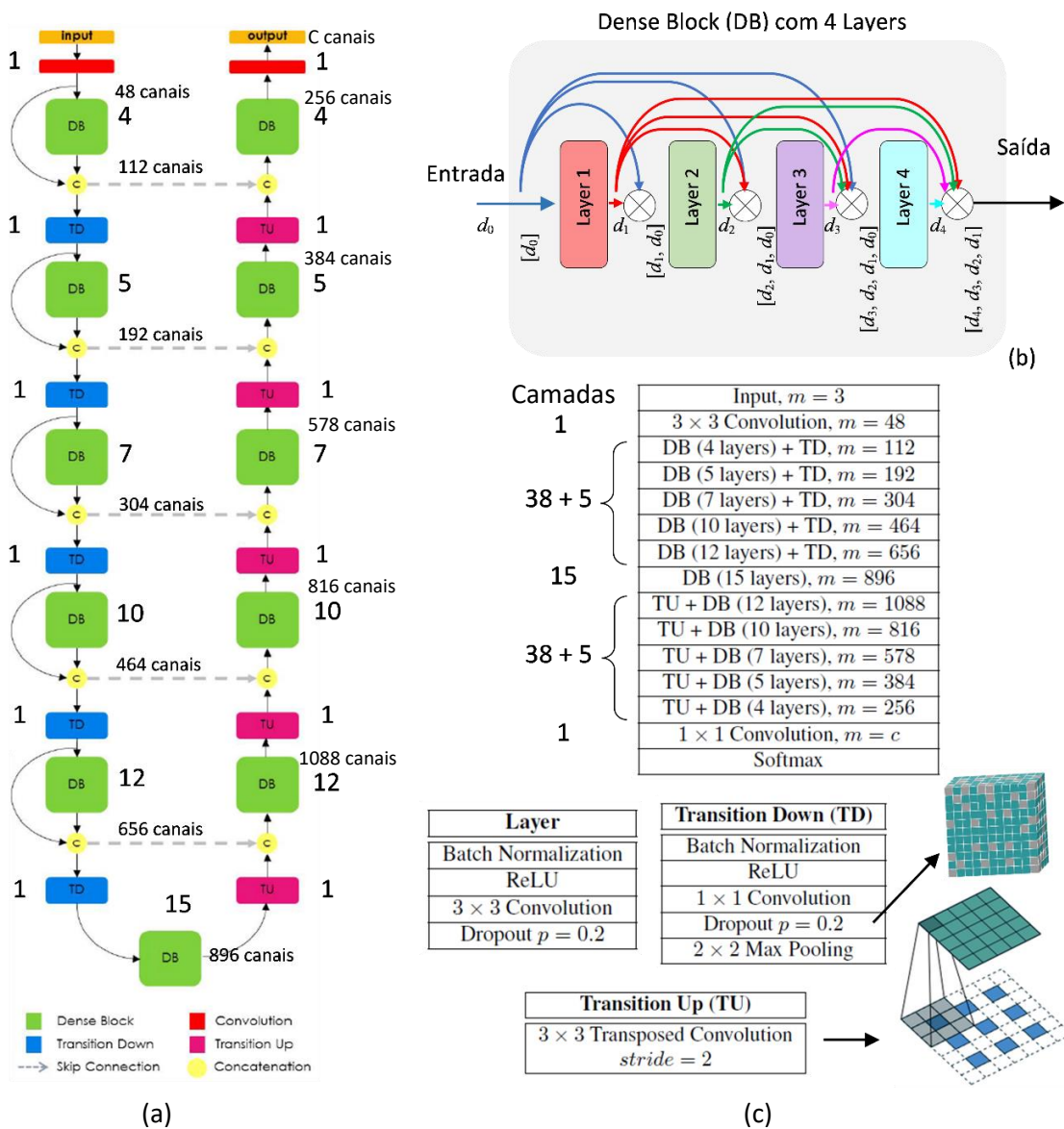
**Figura 2.113** – DenseASPP com taxas de dilatação (3, 6, 12, 18) produz pirâmide de atributos com maior diversidade de escala (em termos de tamanhos de campos receptivos) e maior campo receptivo do que ASPP. FoV representa o tamanho do campo receptivo da combinação densa correspondente [YANG et al. 2018].

A rede DenseASPP [YANG et al. 2018] combina as vantagens de usar camadas de convolução *atrous* em série e em paralelo. A rede de classificação DenseNet [HUANG et al. 2017] pode ser vista como um caso especial de DenseASPP, definindo a taxa de dilatação como 1. Assim, DenseASPP também compartilha as vantagens de DenseNet, incluindo o alívio do problema de desaparecimento de gradiente e menos parâmetros.

### 2.4.10 Modelo FC-DenseNet

A CNN de classificação DenseNet [HUANG *et al.* 2017] é construída a partir de vários blocos densos e camadas de transição, onde cada bloco denso concatena os mapas de atributos de camadas de convolução normal  $3 \times 3$  anteriores e as camadas de transição entre os blocos densos reduzem a dimensionalidade espacial dos mapas de atributos com operações de *pooling*  $2 \times 2$  e passo 2 (Figura 2.49).

Seguindo esta ideia, uma arquitetura de segmentação semântica FC-DenseNet (*Fully Convolutional DenseNet*) [JEGOU *et al.* 2017] foi construída a partir de blocos densos, onde a rede DenseNet totalmente convolucional é estendida, adicionando-se um caminho de *upsampling* ao caminho de *downsampling* para recuperar a resolução de entrada. Desta forma, a rede FC-DenseNet tem uma estrutura codificador-decodificador em forma de U com conexões de salto (Figura 2.114a). Ao contrário da rede DenseASPP, a rede FC-DenseNet não usa convoluções *atrous*.



**Figura 2.114** – Arquitetura da rede FC-DenseNet103 com 103 camadas convolucionais (a), composto dos seguintes módulos: DB (Dense Block) tem diferentes números de camadas (b); TD (Transition Down) e TU (Transition Up) (c);  $m$  corresponde ao número total de mapas de atributos no final de um bloco e  $c$  representa o número de classes [JEGOU *et al.* 2017].

Na [Figura 2.114a](#), uma rede FC-DenseNet103 é composta de um caminho de *downsampling* com cinco blocos densos (*Dense Block* – DB) e cinco módulos *Transition Down* (TD), assim como um caminho de *upsampling* com cinco blocos densos e cinco módulos *Transition Up* (TU). A rede FC-DenseNet103 tem 103 camadas convolucionais: 1 camada na entrada, 38 no caminho de *downsampling*, 15 no bloco denso de gargalo e 38 no caminho de *upsampling*. Há também 5 convoluções de *Transition Down* (TD) e 5 convoluções transposta de *Transition Up* (TU). A camada final é uma convolução  $1 \times 1$  seguida por uma camada *softmax* para fornecer a distribuição por classe em cada *pixel*.

Cada camada (*layer*) de um bloco denso é composta por BN (*Batch Normalization*), seguido por ReLU, uma convolução  $3 \times 3$  com preenchimento “*same padding*” (sem perda de resolução) e *dropout* com probabilidade  $p = 0,2$ . A saída de cada camada  $l$  tem  $k$  mapas de atributos, onde  $k$  é a taxa de crescimento. Na FC-DenseNet103, a taxa de crescimento é definida como  $k = 16$ , portanto, o número de mapas de atributos em cada bloco denso cresce linearmente com a profundidade. Por exemplo, em um bloco denso com 4 camadas ([Figura 2.114b](#)), a primeira camada (*Layer 1*) é aplicada à entrada ( $d_0$ ) para criar  $k$  mapas de atributos ( $d_1$ ), que são concatenados à entrada  $d_0$ . A segunda camada é aplicada à  $[d_1, d_0]$  para criar outros  $k$  mapas de atributos ( $d_2$ ), que são novamente concatenados aos mapas de atributos anteriores ( $[d_2, d_1, d_0]$ ). A operação é repetida quatro vezes. Diferentemente do bloco denso em DenseNet, a saída do bloco denso é a concatenação das saídas das quatro camadas ( $[d_4, d_3, d_2, d_1]$ ) sem a entrada  $d_0$  e, portanto, gera  $4 \times k$  mapas de atributos.

Um módulo TD é introduzido entre os blocos densos para reduzir a dimensionalidade espacial dos mapas de atributos. O módulo TD é composto por BN, seguida por ReLU, uma convolução  $1 \times 1$  (que conserva o número de mapas de atributos), *dropout* com  $p = 0,2$  e uma operação de *maxpooling* não sobreposto de tamanho  $2 \times 2$ , que reduz a resolução do mapa pela metade ([Figura 2.114c](#)).

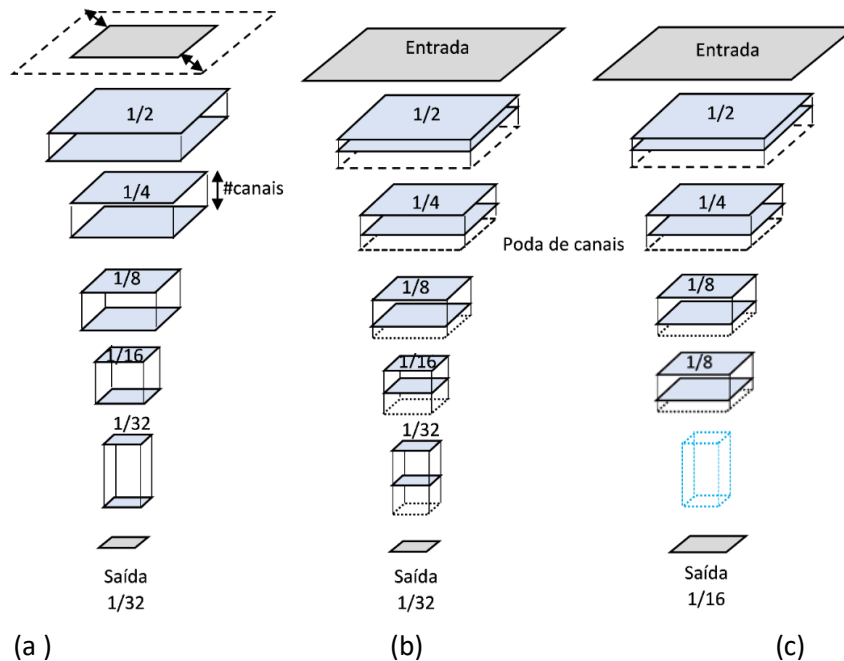
No caminho de *downsampling*, a entrada  $d_0$  para um bloco denso é concatenada com sua saída, levando a um crescimento linear do número de mapas de atributos (compensado pela redução da resolução espacial), enquanto no caminho de *upsampling* não há concatenação da entrada na saída de cada bloco, evitando uma explosão de atributos. O caminho de *upsampling* construído com blocos densos tem um desempenho melhor do que com operações de convolução padrão, como em U-Net.

As conexões de salto dos mapas de maior resolução do caminho de *downsampling* para *upsampling* foram adotadas para permitir uma recuperação de informações refinadas das camadas de *downsampling*. Na [Figura 2.114a](#), o círculo amarelo representa a concatenação e as setas horizontais cinzas representam conexões de salto, sendo que os mapas de atributos do caminho de redução de amostragem são concatenados com os mapas de atributos correspondentes no caminho de aumento de amostragem.

Os módulos TU ([Figura 2.114c](#)) consistem em uma convolução transposta  $3 \times 3$  com passo 2 que faz a sobreamostragem dos mapas de atributos anteriores. Os mapas de atributos sobreamostrados são então concatenados aos provenientes da conexão de salto de mesma resolução, com a finalidade de formar a entrada de um novo bloco denso. A entrada de um bloco denso não é concatenada com sua saída. Assim, a convolução transposta é aplicada apenas aos mapas de atributos obtidos pelo último bloco denso e não a todos os mapas de atributos concatenados até o momento. O último bloco denso resume as informações contidas em todos os blocos densos anteriores na mesma resolução.

### 2.4.11 Modelo BiSeNet

No caso de segmentação semântica em tempo real, existem algumas abordagens para acelerar o modelo: 1) redução do tamanho de entrada para reduzir a complexidade computacional (Figura 2.115a); 2) poda dos canais da rede para aumentar a velocidade de inferência (Figura 2.115b); 3) eliminação das últimas camadas do modelo (Figura 2.115c).



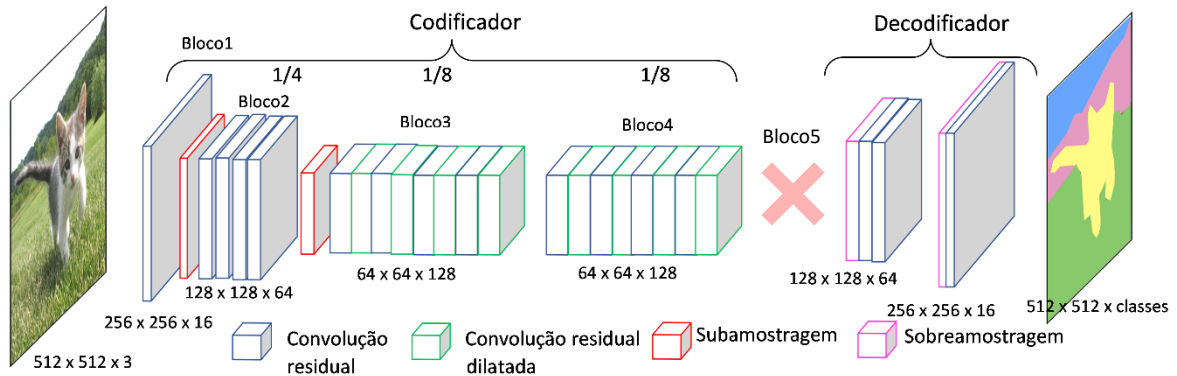
**Figura 2.115** – Modelo de CNN com redução do tamanho de entrada (a); poda dos canais da CNN (b); eliminação do último bloco da CNN (c). A linha tracejada preta representa as operações que danificam a informação espacial, enquanto a linha tracejada azul representa as operações que diminuem o campo receptivo. Adaptado de [YU *et al.* 2018].

Na primeira abordagem (Figura 2.115a), restringir o tamanho da entrada para reduzir a complexidade computacional, cortando ou redimensionando a imagem, gera perda de informações espaciais da imagem original, especialmente nos detalhes, levando à diminuição da precisão que deve ser recuperada pelo modelo de segmentação semântica [YU *et al.* 2018].

Na segunda abordagem (Figura 2.115b), em vez de redimensionar a imagem, são usados *backbones* mais leves devido à poda de canais (*network pruning*). As arquiteturas de segmentação semântica do tipo codificador-decodificador (com/sem conexões de salto), com *backbones* leves no codificador, permitem recuperar a resolução espacial no decodificador; no entanto, esta técnica tem uma desvantagem: informações espaciais perdidas na poda não podem ser facilmente recuperadas [YU *et al.* 2018].

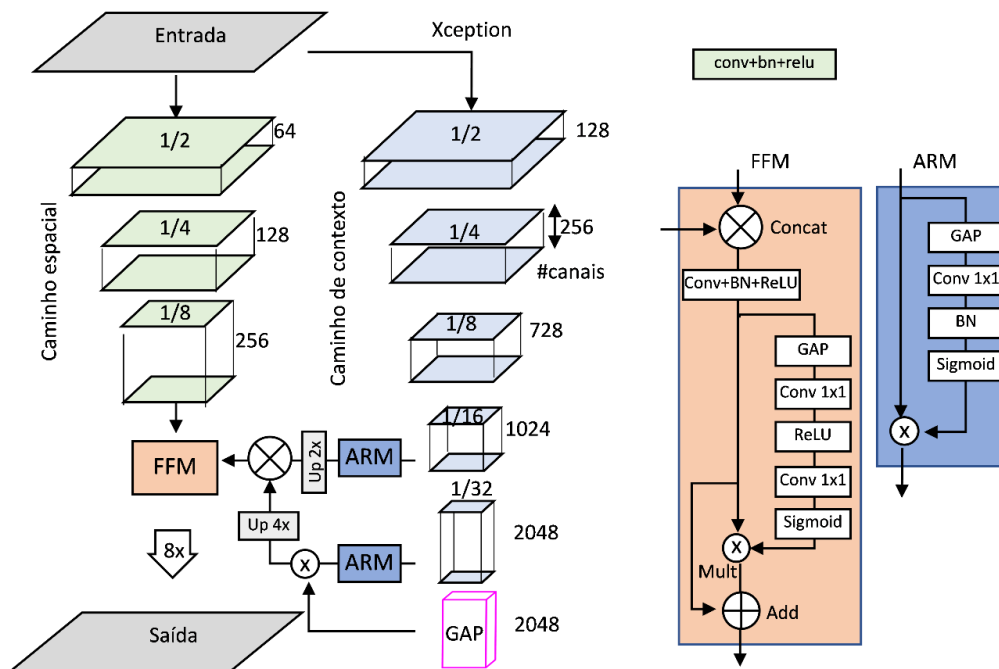
Por outro lado, se os canais forem preservados sem poda de canais e a última camada do codificador for eliminada para preservar a resolução espacial, esta abordagem reduziria a velocidade do modelo de segmentação devido à computação extra em mapas de atributos de alta resolução no decodificador. Da mesma forma, outras arquiteturas com convolução *atrous* ampliam o campo receptivo em múltiplas escalas usando um módulo ASPP com entrada de maior resolução, como DeepLab-v2 e DeepLab-v3 [CHEN *et al.* 2018a; CHEN *et al.* 2017], tendo a desvantagem dessas operações serem computacionalmente exigentes e consumirem memória, o que resulta em baixa velocidade.

Usando a terceira abordagem (Figura 2.115c), uma arquitetura de segmentação semântica mais rápida que SegNet, chamada ENet [PASZKE *et al.* 2016], usa um *backbone* leve, podando os canais para aumentar a velocidade de inferência e eliminando o bloco 5 do *backbone* (Figura 2.116). A rede ENet também elimina a subamostragem no bloco 4 para preservar a resolução espacial com um fator de 8; porém, o campo receptivo não é suficiente para cobrir objetos grandes, sendo necessário usar convoluções residuais dilatadas em série no bloco 4, para obter um maior contexto semântico. Um decodificador leve com menos camadas convolucionais recupera a resolução espacial, sem aumentar a complexidade computacional e tempo de execução.



**Figura 2.116** – Mapas de atributos do modelo ENet com poda de canais e eliminação do último bloco do backbone. Baseado em [PASZKE *et al.* 2016].

Apesar desses métodos indicarem que a informação espacial de alta resolução e campos receptivos maiores são cruciais para alcançar alta precisão, é difícil atender estes requisitos simultaneamente. Para resolver este problema, uma rede de segmentação bilateral (BiSeNet) [YU *et al.* 2018], mostrada na Figura 2.117, contém duas partes: um caminho espacial (caixas verdes) e um caminho de contexto (caixas azul-claras).



**Figura 2.117** – Estrutura da rede de segmentação bilateral (BiSeNet) com caminho espacial e caminho de contexto. Adaptado de [YU *et al.* 2018].



O caminho espacial foi projetado para preservar as informações espaciais da imagem original, contendo um modelo leve com apenas três camadas de convolução. Cada camada inclui uma convolução  $3 \times 3$  com passo 2, seguida de normalização em lote (BN) e ReLU. Portanto, esse caminho gera os mapas de atributos de alta resolução (saída com resolução de  $1/8$  da imagem original), codificando informações espaciais ricas devido à maior resolução espacial dos mapas de atributos.

O caminho de contexto utiliza um *backbone* leve, por exemplo, Xception pré-treinado [CHOLLET 2017], para reduzir rapidamente o mapa de atributos e obter um grande campo receptivo que codifica informações de contexto semântico de alto nível. Outras CNN-base também foram usadas, como a rede ResNet-18 e ResNet-101 [HE et al. 2016a]. Em seguida, um agrupamento de média global (GAP) é anexado no final do *backbone*, para obter um campo receptivo máximo com informações de contexto global.

Uma estrutura de salto foi usada para fundir os mapas de atributos dos dois últimos estágios do *backbone*, refinados pelo módulo ARM (*Attention Refinement Module*), com a saída do agrupamento global sobreamostrada.

Na Figura 2.118, as informações espaciais capturadas pelo mapa de atributos de baixo nível na saída do caminho espacial codificam informações ricas em detalhes, enquanto o mapa de atributos de alto nível na saída do caminho de contexto codifica informações de contexto. Portanto, a rede BiSeNet usa um módulo FFM (*Feature Fusion Module*) para combinar os atributos de saída desses dois caminhos. Por fim, como a resolução da previsão final é  $1/8$  da imagem original, ela é sobreamostrada em 8x. Com este modelo leve, a rede BiSeNet pode alcançar desempenho da velocidade em tempo real e alta precisão da segmentação semântica simultaneamente.

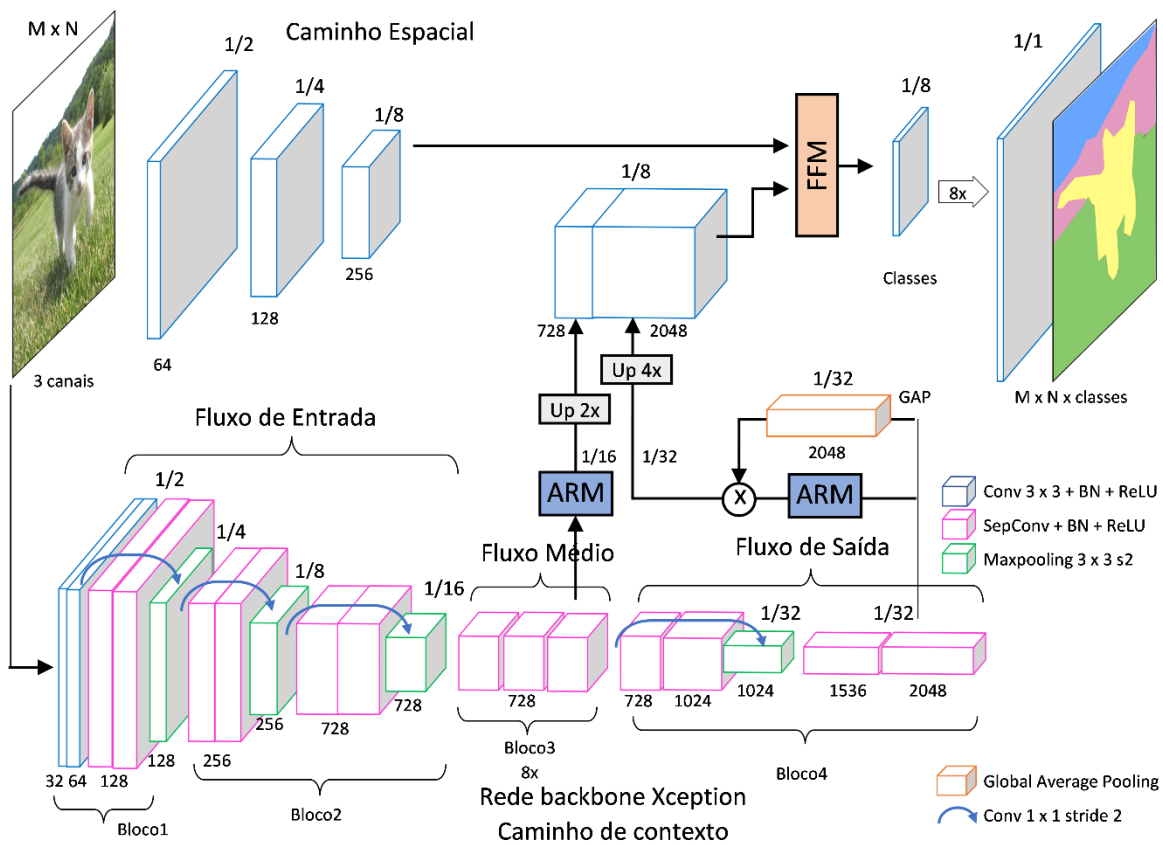
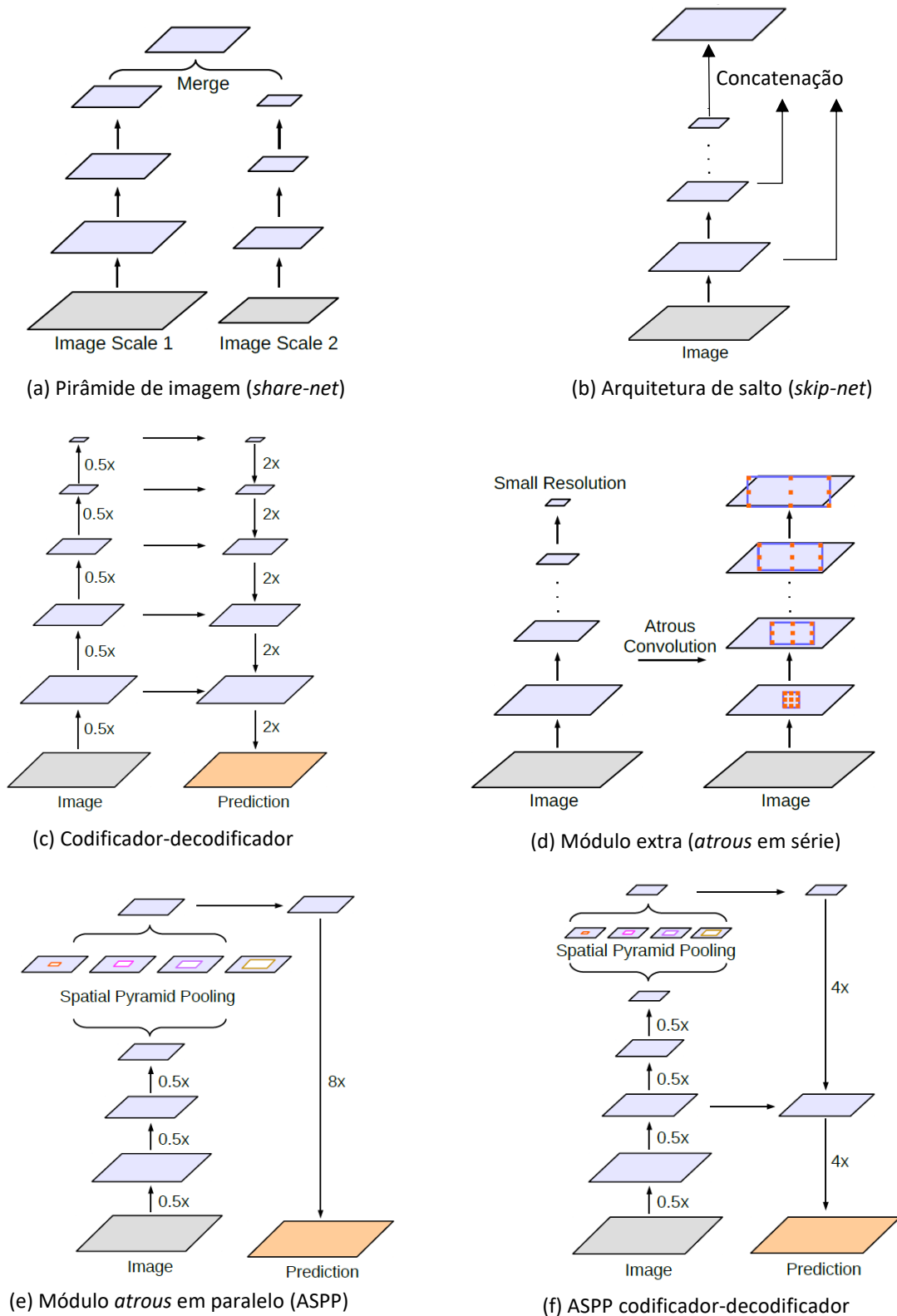


Figura 2.118 – Mapas de atributos da rede de segmentação bilateral (BiSeNet).

### 2.4.12 Comparações entre redes

As redes de segmentação semântica descritas nas seções anteriores podem ser divididas em algumas categorias, como mostra a [Figura 2.119](#).



**Figura 2.119** – Diferentes estruturas de redes de segmentação semântica [CHEN *et al.* 2017; CHEN *et al.* 2018b].



(a) Arquitetura de pirâmide de imagem

Algumas redes pertencentes a esta categoria (Figura 2.119a) – como [FARABET *et al.* 2013], Piecewise [LIN *et al.* 2016] e Attention to Scale [CHEN *et al.* 2016a] – extraem atributos multiescala, redimensionando a imagem de entrada para várias escalas (ou seja, pirâmide de imagem), que são aplicadas a uma rede compartilhada (*share-net*) para explorar as informações contextuais para previsão da segmentação. Assim, o mesmo modelo de rede com pesos compartilhados é aplicado a entradas multiescalas. Em seguida, a rede calcula um mapa de previsão para classificação *pixel a pixel*. Os mapas de atributos dos ramos paralelos da CNN são interpolados bilinearmente para a resolução da imagem original e mesclados em mapas de atributos multiescala. As respostas de mapas de atributos das entradas de pequena escala codificam o contexto de longo alcance (contexto global), enquanto as entradas de grande escala preservam os detalhes de pequenos objetos (contexto local).

Para explorar entradas multiescala, a rede Attention to Scale [CHEN *et al.* 2016a] na Figura 2.120 adapta o modelo DeepLab-LargeFOV (v1) [CHEN *et al.* 2015], com parâmetros de rede inicializados a partir do modelo VGG-16 pré-treinado no ImageNet [SIMONYAN e ZISSERMAN 2015]. Nesta rede, um modelo de atenção aprende a ponderar suavemente os atributos de diferentes escalas de entrada para prever o rótulo semântico de um *pixel*, sendo que a saída final é produzida executando uma operação *softmax* na soma ponderada dos mapas de pontuação em todas as escalas.

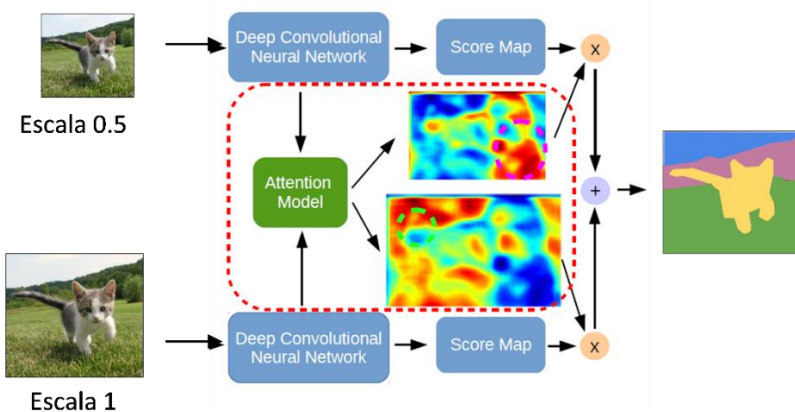


Figura 2.120 – Estrutura da rede Attention to Scale [CHEN *et al.* 2016a].

(b) Arquitetura de salto

A arquitetura de rede com conexão de salto (*skip-net*) combina atributos das camadas intermediárias para produzir a saída de classificação final de *pixels*, por meio de adição ou concatenação dos mapas de atributos (Figura 2.119b). As redes Hypercolumns [HARIHARAN *et al.* 2014a], FCN [LONG *et al.* 2015], DeepLab-MSc (v1) [CHEN *et al.* 2015] e BiSeNet [YU *et al.* 2018] fazem parte desta categoria.

A rede Hypercolumns na Figura 2.58 classifica um *pixel* com representação em hipercoluna (ou seja, concatenação de atributos de camadas intermediárias sobreamostradas). A rede totalmente convolucional FCN, mostrada na Figura 2.67, combina os mapas de atributos de camadas intermediárias por meio de conexões de salto e utiliza deconvolução com convoluções transpostas, que aprendem a aumentar progressivamente a amostragem (*upsampling*), produzindo previsões de *pixels* na

resolução de entrada. A rede DeepLab-MSc (v1), mostrada na [Figura 2.92](#), aplica redes *MultiLayer Perceptrons* (MLPs) à imagem de entrada e às saídas das quatro primeiras camadas de *pooling* a fim de extrair atributos multiescala. Os mapas de atributos resultantes são concatenados ao mapa de atributos da última camada da rede principal por meio de conexões de salto. A rede BiSeNet, na [Figura 2.117](#), combina mapas de atributos de baixo nível e maior resolução do caminho espacial com informações de contexto semântico de alto nível do caminho de contexto.

### (c) Arquitetura codificador-decodificador

Estas arquiteturas totalmente convolucionais contêm módulos codificador-decodificador (*encoder-decoder*), com ou sem conexões de salto ([Figura 2.119c](#)). O codificador reduz gradualmente a dimensão espacial dos mapas de atributos e, portanto, as informações semânticas de longo alcance (maior contexto) são capturadas na saída do codificador. O decodificador recupera gradualmente as informações espaciais e os detalhes do objeto, aprendendo por meio de *upsampling* e de uma série de convoluções a mapear os mapas de atributos de baixa resolução (da camada mais profunda do codificador) para rótulos semânticos com as dimensões da imagem de entrada.

Assim, as redes codificador-decodificador sem conexões de salto, como DeconvNet [[NOH et al. 2015](#)] e SegNet-v2 [[BADRINARAYANAN et al. 2017](#)], podem capturar limites de objetos mais nítidos, recuperando gradualmente as informações espaciais no decodificador. Por exemplo, a rede DeconvNet na [Figura 2.75](#) emprega *upsampling* usando índices de *pooling*, com intuito de aprender a deconvolução de atributos de baixa para alta resolução, usando convolução transposta. A rede SegNet-v2 na [Figura 2.78](#) utiliza os índices de *maxpooling* do codificador e aprende camadas convolucionais extras para densificar as respostas dos atributos.

Por outro lado, as redes com estrutura codificador-decodificador com conexões de salto, como U-Net [[RONNEBERGER et al. 2015](#)], RefineNet [[LIN, MILAN et al. 2017](#)] e FC-DenseNet [[JEGOU et al. 2017](#)], concatenam mapas de atributos do codificador para refinamento dos mapas de atributos do decodificador. Outras redes, como LRR [[GHIASI e FOWLKES 2016](#)], FRRN [[POHLEN et al. 2017](#)] e LargeKernel [[PENG et al. 2017](#)], também demonstraram a eficácia de modelos baseados na estrutura codificador-decodificador com conexões de salto para segmentação semântica.

### (d) Arquitetura de módulo extra com convolução *atrous* em série

Esta arquitetura usa um módulo extra no topo de uma rede *frontend* de segmentação para capturar informações de longo alcance (maior campo receptivo e campo de visão) ([Figura 2.119d](#)). Por exemplo, a rede DilatedNet na [Figura 2.86](#) emprega um módulo de contexto com várias camadas convolucionais *atrous* em série para aumentar o campo receptivo e capturar gradualmente informações de maior contexto. A rede DeepLab-v3 em série na [Figura 2.99](#) replica três vezes a última camada do *backbone* e adiciona convoluções extras com múltiplas taxas de dilatação. Em vez de adicionar um módulo de contexto, as redes DeepLab-v1 básico na [Figura 2.91](#) e DeepLab-LargeFoV (v1) na [Figura 2.93](#) usam a convolução *atrous* em série nos mapas de atributos das camadas finais do *backbone* (e não nos  $C$  mapas de previsão, como ocorre na rede DilatedNet), para capturar um maior contexto adotando taxas crescentes de dilatação.

(e) Arquitetura com convolução *atrous* em paralelo (ASPP)

As arquiteturas de pirâmide espacial de *pooling*, tal como a rede PSPNet (*Pyramid Scene Parsing Net*) [ZHAO *et al.* 2017] na Figura 2.103, realizam agrupamento espacial em várias escalas de grade. Já as redes DeepLab-v2 com *backbone* VGG (Figura 2.95) e ResNet (Figura 2.97) usam uma pirâmide espacial ASPP com convolução *atrous*, onde camadas de convoluções *atrous* paralelas com diferentes taxas de dilatação capturam informações multiescala com campos de visão diferentes. A rede DeepLab-v3 com *backbone* ResNet na Figura 2.104 [CHEN *et al.* 2017] adiciona ao ASPP um atributo de nível de imagem que codifica o contexto global e aumenta ainda mais o desempenho.

Nas arquiteturas que usam ASPP, os mapas de atributos extraídos em cada ramo paralelo são fundidos (somados ou concatenados) para gerar mapas de atributos em poucas escalas. Por outro lado, a rede DenseASPP [YANG *et al.* 2018] na Figura 2.110 aplica um módulo extra DenseASPP para capturar atributos multiescala de forma densa, utilizando um módulo de convoluções *atrous* em série e em paralelo conectadas por conexões densas.

(f) Arquitetura com módulo ASPP codificador-decodificador

A rede DeepLab-v3+ [CHEN *et al.* 2018b] na Figura 2.106 usa a rede DeepLab-v3 com ASPP no codificador e adiciona um módulo decodificador para refinar os atributos.

### 2.4.13 Desafios da segmentação semântica

Existem dois desafios principais na aplicação de CNNs originalmente projetadas para classificação de imagens, como VGG [SIMONYAN e ZISSERMAN 2015] e ResNet [HE *et al.* 2016a], em tarefas de segmentação semântica de imagens: (1) redução da resolução espacial de atributos, causada por camadas consecutivas de *maxpooling*; e (2) objetos em várias escalas na imagem. Cada tipo de arquitetura de rede de segmentação semântica de imagens aborda estes problemas de diferentes formas.

#### 1) Redução da resolução espacial

O primeiro desafio é a baixa resolução dos mapas de atributos, causada por operações de subamostragem (*pooling*) consecutivas ou convolução com passos maiores que 1, o que permite que as CNNs aprendam representações de atributos cada vez mais abstratas com maior campo receptivo. No entanto, a segmentação semântica de imagens requer atributos de alto nível para representar cada *pixel* em uma previsão semântica.

Para compensar a baixa resolução de atributos de alto nível, as arquiteturas de salto, como a rede totalmente convolucional (FCN) [LONG *et al.* 2015; SHELHAMER *et al.* 2017], usam camadas de *upsampling* combinadas com conexões de salto de mapas de atributos intermediários para recuperar a resolução original [CHEN *et al.* 2017].

As arquiteturas codificador-decodificador utilizam: 1) *unpooling* com conexões de atalho (usando índices de *pooling*) em DeconvNet e SegNet [BADRINARAYANAN *et al.* 2017] para recuperar os detalhes do objeto nas camadas de deconvolução; ou 2) *upsampling* com interpolação bilinear ou convolução transposta, combinada com conexões de salto de mapas de maior resolução das camadas intermediárias, em U-Net [RONNEBERGER *et al.* 2015], RefineNet [LIN, MILAN *et al.* 2017] e FC-DenseNet [JEGOU *et al.* 2017].

As arquiteturas que usam convoluções dilatadas/*atrous* em série ou em paralelo (ASPP), como as redes DilatedNet [YU e KOLTUN 2016], DeepLab-v1, v2, v3 e v3+ [CHEN *et al.* 2015; CHEN *et al.* 2018a; CHEN *et al.* 2017; CHEN *et al.* 2018b], além de DenseASPP [YANG *et al.* 2018], removem as operações de subamostragem das últimas camadas para preservar o tamanho espacial do mapa de atributos. Elas também empregam a convolução *atrous*, que permite ampliar efetivamente o campo de visão dos filtros sem redução da resolução espacial.

Já a rede BiSeNet [YU *et al.* 2018] usa um caminho espacial separado para preservar as informações espaciais da imagem original e conexões de salto para concatenar mapas de atributos intermediários do caminho de contexto.

## 2) Atributos multiescala

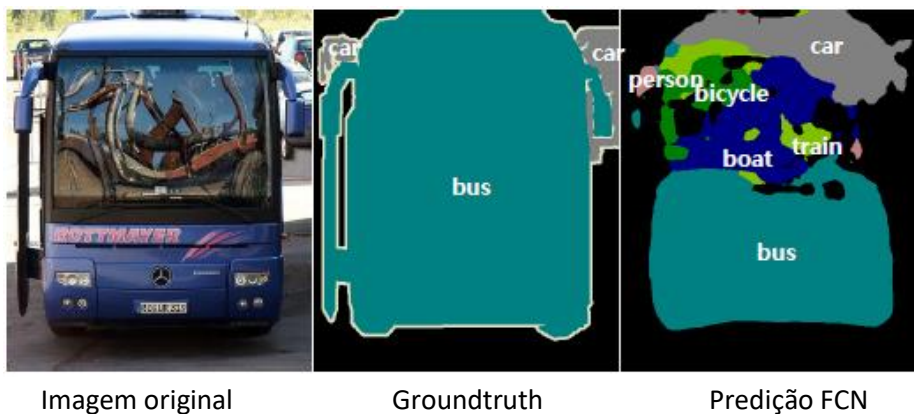
Em determinadas aplicações como, por exemplo, direção autônoma com tarefa de segmentação de cenas externas de estradas, a rede deve ter a capacidade de classificar *pixels* de objetos grandes, como estradas e edifícios, mas também delinear objetos pequenos, como carros, bicicletas e pedestres, em diferentes escalas de *zoom*. Além disso, em imagens de alta resolução é necessário um maior contexto. Para lidar com este desafio de segmentação de objetos em múltiplas escalas, os mapas de atributos devem ser capazes de cobrir diferentes tamanhos de campos receptivos. Para isso, as arquiteturas de rede usam diferentes estratégias para capturar contexto multiescala.

A primeira estratégia usada nas arquiteturas de pirâmide de imagem apresenta versões redimensionadas da mesma imagem à rede e, em seguida, agrega os mapas de atributos para extrair informações de objetos em várias escalas [CHEN *et al.* 2016a].

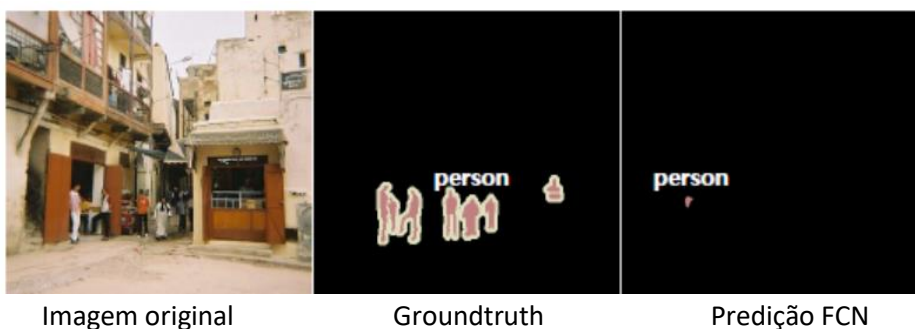
As arquiteturas de rede com conexões de salto, como FCN, geralmente falham na previsão de objetos de diferentes tamanhos. Portanto, um objeto substancialmente maior ou menor do que o campo receptivo pode ser fragmentado ou mal rotulado. Em outras palavras, a previsão do rótulo é feita com informações locais (campos receptivos pequenos) para objetos grandes e os *pixels* que pertencem ao mesmo objeto podem ter rótulos inconsistentes, conforme mostrado na Figura 2.121a. Por outro lado, um objeto substancialmente menor do que o campo receptivo é frequentemente ignorado e classificado como fundo (Figura 2.121b). Embora as redes tentem contornar essa limitação usando uma arquitetura de salto com mapas de atributos em diferentes resoluções, esta solução não apresenta um ganho de desempenho significativo para objetos multiescala.

Para tratar as limitações de FCN em rotular objetos multiescala, a rede DeconvNet realiza a segmentação semântica por instância de objetos na imagem (subimagens com instâncias individuais de objetos de escalas diferentes) para obter mapas de segmentação semântica de propostas individuais, que são agregadas para produzir segmentação semântica de uma imagem inteira.

As arquiteturas codificador-decodificador com ou sem conexão de salto, como U-Net, RefineNet e SegNet não exploram atributos multiescala devido ao seu campo receptivo de tamanho fixo. Por exemplo, a SegNet-v1 tem quatro camadas de convolução e *maxpooling* no codificador, obtendo um contexto espacial fixo de  $106 \times 106$  *pixels*. A adição de mais camadas apenas aumenta o contexto espacial de escala única na imagem de entrada, com campo receptivo de tamanho fixo, mas geram previsões mais suaves nos limites dos objetos à medida que a profundidade aumenta.



(a) Rótulos inconsistentes devido ao grande tamanho do objeto e menor campo receptivo.



(b) Rótulos ausentes devido ao pequeno tamanho do objeto e maior campo receptivo.

**Figura 2.121** – Limitações dos algoritmos de segmentação semântica com conexões de salto sem atributos multiescala [NOH *et al.* 2015].

As arquiteturas com um módulo extra no topo de uma rede de segmentação semântica, como DilatedNet [YU e KOLTUN 2016] e DeepLab-v3 em série, usam um módulo de contexto composto de várias camadas de convoluções *atrous* consecutivas com taxas de dilatação e campos receptivos crescentes, com objetivo de capturar informações de contexto de longo alcance e obter maior campo de visão, sem diminuir a resolução espacial. No entanto, apesar de obter tamanhos de campo receptivo diferentes em cada camada, os neurônios no mapa de atributos de saída da rede têm um campo de visão de tamanho fixo em escala única, não sendo possível agregar atributos multiescala.

Motivado pela pirâmide espacial de *pooling* em SPPNet [HE *et al.* 2015b], as arquiteturas de rede DeepLab-v2, DeepLab-v3 e v3+ usam um módulo ASPP com quatro camadas de convoluções *atrous* paralelas de diferentes taxas de dilatação e diferentes campos de visão, capturando informações de contexto em várias escalas. Já a rede DenseASPP [YANG *et al.* 2018] combina as vantagens de usar camadas de convolução *atrous* em paralelo e em série, para compor simultaneamente uma pirâmide de atributos muito mais densa com várias escalas.

A Tabela 2.4 mostra um comparativo das características das redes de segmentação semântica descritas nesta seção, como também os resultados de desempenho nos conjuntos de dados PASCAL-VOC 2012 e Cityscapes, como referido no artigo original de cada rede. O desempenho da rede de segmentação é medido em termos de média de intersecção sobre união (mIoU) entre o *groundtruth* e o mapa de previsão de classe, descrito no Apêndice C. Os melhores resultados foram obtidos pelas redes multiescala, como PSPNet, DeepLab-v3+ e DenseASPP.



**Tabela 2.4 – Tabela comparativa de características e desempenho (mIoU%) das redes de segmentação semântica no conjunto PASCAL-VOC e Cityscapes.**

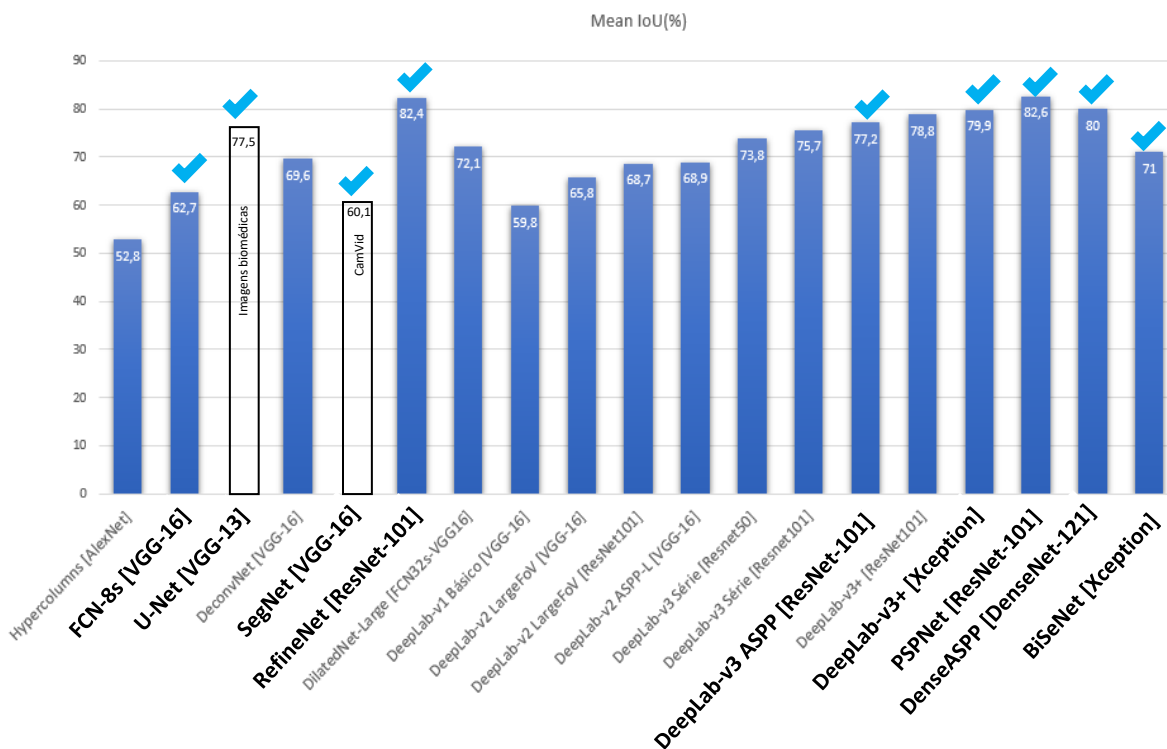
Rede de Segmentação Semântica	CNN-base (backbone)	Deconvolução										Validação Pascal-VOC	Teste Pascal-VOC	Valid. Cityscapes	Teste Cityscapes					
		Fully convolutional	CRF	Upsampling final	Interpolação Bilinear	Transposta (stride s)	Conexão de Salto	Encoder/decoder	Índice de pooling	Módulo Contexto	Convolução atrous em série					Convolução atrous em paralelo ASPP	Contexto Global	Pré-treinamento	Aumento dados	
Hypercolumns	AlexNet	-	-	8x	✓	-	✓	-	-	-	-	-	-	-	-	52,8	-	-	-	
FCN32s	AlexNet	✓	-	32x	✓	(s=32)	-	-	-	-	-	-	-	-	✓	-	39,8	-	-	-
	VGG-16	✓	-	32x	✓	(s=32)	-	-	-	-	-	-	-	-	✓	-	56,0	-	-	-
	GoogLeNet	✓	-	32x	✓	(s=32)	-	-	-	-	-	-	-	-	✓	-	42,5	-	-	-
FCN32s	VGG-16	✓	-	32x	✓	(s=32)	-	-	-	-	-	-	-	-	✓	✓	59,4	-	-	-
FCN16s	VGG-16	✓	-	16x	✓	(2,16)	✓	-	-	-	-	-	-	-	✓	✓	62,4	-	-	-
<b>FCN8s</b>	<b>VGG-16</b>	<b>✓</b>	<b>-</b>	<b>8x</b>	<b>✓</b>	<b>(2,2,8)</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>62,7</b>	<b>62,2</b>	<b>-</b>	<b>65,3</b>
<b>U-Net</b>	<b>VGG-13</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>
DeconvNet	VGG-16	-	-	-	-	(s=1)	-	✓	✓	-	-	-	-	-	✓	✓	-	69,6	-	-
DeconvNet+CRF	VGG-16	-	✓	-	-	(s=1)	-	✓	✓	-	-	-	-	-	✓	✓	-	70,5	-	-
EDeconvNet	VGG-16	-	-	-	-	(s=1)	-	✓	✓	-	-	-	-	-	✓	✓	-	71,7	-	-
EDeconvNet+CRF	VGG-16	-	✓	-	-	(s=1)	-	✓	✓	-	-	-	-	-	✓	✓	-	72,5	-	-
SegNet v1	4 camadas 7x7	✓	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
<b>SegNet v2</b>	<b>VGG-16</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>
<b>RefineNet-Res50</b>	<b>ResNet-50</b>	<b>✓</b>	<b>-</b>	<b>4x</b>	<b>✓</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>
RefineNet-Res101	ResNet-101	✓	-	4x	✓	-	✓	✓	-	-	-	-	-	-	✓	✓	-	82,4	-	73,6
RefineNet-Res152	ResNet-152	✓	-	4x	✓	-	✓	✓	-	-	-	-	-	-	✓	✓	-	83,4	-	-
DilatedNet	Basic (C canais)	FCN32s-VGG16	✓	-	8x	-	-	-	-	✓	(2,4) + (2,4,8,16)	-	-	-	✓	-	71,3	-	-	-
	Large (> C canais)	FCN32s-VGG16	✓	-	8x	-	-	-	-	✓	(2,4) + (2,4,8,16)	-	-	-	✓	-	72,1	73,5	-	-
	Basic + CRF	FCN32s-VGG16	✓	✓	8x	-	-	-	-	✓	(2,4) + (2,4,8,16)	-	-	-	✓	-	72,7	-	-	-
	Large + CRF	FCN32s-VGG16	✓	✓	8x	-	-	-	-	✓	(2,4) + (2,4,8,16)	-	-	-	✓	-	73,3	74,7	-	-
DeepLab v1 Básico	VGG-16	✓	-	8x	-	-	-	-	-	-	(2) + (r=4)	-	-	-	✓	✓	59,8	-	-	-
	CRF	VGG-16	✓	✓	8x	-	-	-	-	-	(2) + (r=4)	-	-	-	✓	✓	63,7	66,4	-	-
	MSC-CRF	VGG-16	✓	✓	8x	-	-	✓	-	-	(2) + (r=4)	-	-	-	✓	✓	65,2	67,1	-	-
	CRF (7x7)	VGG-16	✓	✓	8x	-	-	-	-	-	(2) + (r=4)	-	-	-	✓	✓	67,6	70,3	-	-
	*MSC + CRF + LargeFov	VGG-16	✓	✓	8x	-	-	✓	-	-	(2) + (r=12)	-	-	-	✓	✓	68,7	71,6	-	-
DeepLab v2	LargeFov	VGG16	✓	-	8x	-	-	-	-	-	(2) + (r=12)	-	-	-	✓	✓	65,8	-	-	-
	ASPP-S	VGG16	✓	-	8x	-	-	-	-	-	(2)	r = {6,12,18,24}	-	-	✓	✓	66,9	-	-	-
	ASPP-L	VGG16	✓	-	8x	-	-	-	-	-	(2)	r = {2,4,8,12}	-	-	✓	✓	68,9	-	-	-
	LargeFov + CRF	VGG16	✓	✓	8x	-	-	-	-	-	(2) + (r=12)	-	-	-	✓	✓	69,8	-	-	-
	ASPP-S + CRF	VGG16	✓	✓	8x	-	-	-	-	-	(2)	r = {6,12,18,24}	-	-	✓	✓	69,7	-	-	-
	ASPP-L + CRF	VGG16	✓	✓	8x	-	-	-	-	-	(2)	r = {2,4,8,12}	-	-	✓	✓	71,6	72,7	-	63,1
	LargeFov	ResNet101	✓	-	8x	-	-	-	-	-	(2) + (r=12)	-	-	-	✓	-	68,7	-	-	-
	MSC + LargeFov	ResNet101	✓	-	8x	-	-	-	-	-	(2) + (r=12)	-	-	-	✓	✓	75,5	-	-	-
	MSC + ASPP-L	ResNet101	✓	-	8x	-	-	-	-	-	(2,4)	r = {6,12,18,24}	-	-	✓	✓	76,3	-	-	-
	MSC+ASPP-L + CRF	ResNet101	✓	✓	8x	-	-	-	-	-	(2,4)	r = {6,12,18,24}	-	-	✓	✓	77,7	79,7	71,4	70,4
DeepLab v3 (Série)	ResNet-50	✓	-	16x	-	-	-	-	-	✓	(2) + (4,8,16)	-	-	-	✓	-	73,8	-	-	-
	Série	ResNet-101	✓	-	16x	-	-	-	-	✓	(2) + (4,8,16)	-	-	-	✓	-	75,7	-	-	-
	ASPP	ResNet-101	✓	-	16x	-	-	-	-	✓	(2)	r = {6,12,18}	✓	✓	✓	✓	77,2	85,7	-	-
<b>DeepLab-v3+ (ASPP)</b>	ResNet-101	✓	-	4x	✓	-	✓	✓	-	✓	(2)	r = {6,12,18}	✓	✓	✓	✓	78,8	-	-	-
	<b>Xception mod.</b>	<b>✓</b>	<b>-</b>	<b>4x</b>	<b>✓</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>✓</b>	<b>(2)</b>	<b>r = {6,12,18}</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>79,9</b>	<b>87,8</b>	<b>-</b>	<b>-</b>
<b>DenseASPP</b>	DenseNet-121	✓	-	8x	-	-	✓	-	-	✓	(6,12)	r = {6,12}	-	-	✓	✓	-	-	74,8	-
	DenseNet-121	✓	-	8x	-	-	✓	-	-	✓	(6,12,18)	r = {6,12,18}	-	-	✓	✓	-	-	75,6	-
	DenseNet-121	✓	-	8x	-	-	✓	-	-	✓	(3,6,12,18)	r = {3,6,12,18}	-	-	✓	✓	-	-	75,7	-
	DenseNet-121	✓	-	8x	-	-	✓	-	-	✓	(6,12,18,24)	r = {6,12,18,24}	-	-	✓	✓	-	-	76,2	-
	DenseNet-169	✓	-	8x	-	-	✓	-	-	✓	(6,12,18,24)	r = {6,12,18,24}	-	-	✓	✓	-	-	77,7	-
	DenseNet-201	✓	-	8x	-	-	✓	-	-	✓	(6,12,18,24)	r = {6,12,18,24}	-	-	✓	✓	-	-	78,9	-
	<b>DenseNet-121</b>	<b>✓</b>	<b>-</b>	<b>8x</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>{3,6,12,18,24}</b>	<b>r = {3,6,12,18,24}</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>76,6</b>	<b>80,6</b>
	DenseNet-121	✓	-	8x	-	-	✓	-	-	✓	(3,6,12,18,24,30)	{3,6,12,18,24,30}	-	-	✓	✓	-	-	76,5	-
<b>BiSeNet (FCN-32s)</b>	<b>Xception39</b>	<b>✓</b>	<b>-</b>	<b>8x</b>	<b>✓</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>-</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>-</b>	<b>-</b>	<b>72,0</b>	<b>71,4</b>
	ResNet-18	✓	-	8x	✓	-	✓	✓	-	-	-	-	-	✓	✓	✓	-	-	78,6	77,7
	ResNet-101	✓	-	8x	✓	-	✓	✓	-	-	-	-	-	✓	✓	✓	-	-	80,3	78,9
PSPNet	ResNet-101	✓	-	-	-	-	-	-	-	✓	-	-	-	✓	✓	✓	-	82,6	-	78,4

\*DeepLab-LargeFov, MSC – MultiScale, CRF – Conditional Random Field, ASPP – Atrous Spatial Pyramid Pooling, FoV – Field of View

A [Figura 2.122](#) mostra o gráfico de desempenho de previsão de algumas redes usando o conjunto PASCAL-VOC 2012, conforme relatado nos artigos originais. Os valores apresentados correspondem ao desempenho mIoU de teste das redes criadas em diferentes épocas, que utilizam estratégias de treinamento e teste parecidas, por exemplo: aumento de dados durante o treinamento com redimensionamento/recorte aleatório e inversão horizontal das imagens; treinamento e avaliação em várias escalas, com a média das previsões em diferentes escalas para previsão final, etc. Mesmo assim, a comparação não é ideal devido às diferenças existentes na configuração do treinamento e teste.

As primeiras redes de conexão de salto Hypercolumns [[HARIHARAN et al. 2014a](#)] e FCN obtiveram os menores desempenhos. Algumas redes estruturadas como codificador-decodificador foram avaliadas em outro conjunto, não permitindo uma comparação direta de desempenho, como a rede U-Net [[RONNEBERGER et al. 2015](#)] avaliada em um conjunto de dados biomédico específico e SegNet-v2 [[BADRINARAYANAN et al. 2017](#)] avaliada no conjunto de dados CamVid na tarefa de segmentação de cenas de estradas externas em aplicações de direção autônoma. A rede RefineNet [[LIN, MILAN et al. 2017](#)] superou DeepLab-v2, com desempenho muito próximo do DeepLab-v3 e DeepLab-v3+ [[CHEN et al. 2017](#); [CHEN et al. 2018b](#)]. O modelo DeepLab-v3+ teve um melhor desempenho que versões anteriores do DeepLab, sem qualquer pós-processamento CRF. As redes com atributos multiescala, como PSPNet [[ZHAO et al. 2017](#)], DeepLab-v3+ e DenseASPP [[YANG et al. 2018](#)] obtiveram melhor desempenho que BiSeNet [[YU et al. 2018](#)]. Porém, a rede BiSeNet é mais leve, com menor tempo de inferência.

As redes destacadas em negrito serão avaliadas no [Capítulo 5](#), usando as mesmas estratégias de treinamento e teste em nosso conjunto de dados de domínio específico, com intuito de garantir as mesmas condições para comparação de desempenho das nove redes de segmentação semântica selecionadas para uma aplicação de agricultura.



**Figura 2.122** – Comparação de desempenho de segmentação no conjunto PASCAL-VOC.



## 2.5 Transferência de aprendizado

Na prática, o treinamento de uma CNN profunda do zero, com inicializações aleatórias de pesos, exige uma grande quantidade de imagens para evitar sobreajuste da rede. Desta forma, uma estratégia muito utilizada para treinamento de grandes redes em pequenos conjuntos de dados é a transferência de aprendizado [CHOLLET 2016].

No aprendizado por transferência, os pesos de uma CNN-base – pré-treinada em uma tarefa de classificação de imagens em um conjunto de dados de referência de larga escala, por exemplo, um subconjunto ImageNet que possui mais de 1 milhão de imagens e 1.000 classes – são transferidos para inicializar uma rede-destino a ser treinada em um conjunto de dados e tarefa específicos (classificação, segmentação, etc.) [OQUAB *et al.* 2014].

O processo de transferência de aprendizado funciona melhor se, em vez de específicos da tarefa-base, os atributos da rede-base forem gerais o suficiente para outras tarefas de visão computacional com características diferentes da original. Segundo Yosinski *et al.*, o nível de generalidade e, portanto, de reutilização das representações extraídas em cada camada da CNN, dependem da profundidade da camada na hierarquia de atributos [YOSINSKI *et al.* 2014]. As primeiras camadas extraem mapas de atributos gerais (como arestas, cores e texturas), enquanto as camadas superiores extraem conceitos mais abstratos e representam atributos específicos para um determinado conjunto de dados ou tarefa. No entanto, entre a primeira camada de atributos gerais e a última camada de atributos específicos da rede pré-treinada, há uma transição de geral para específico em alguma camada intermediária. Assim, se o novo conjunto de dados for muito diferente do conjunto de base, apenas as primeiras camadas convolucionais do modelo devem ser usadas para transferência de aprendizado. Na prática, geralmente as primeiras  $n$  camadas da rede-base são copiadas para as primeiras  $n$  camadas da rede-destino e as camadas restantes da rede-destino são inicializadas aleatoriamente.

Existem duas técnicas para reutilização de uma rede CNN pré-treinada: extração de atributos ou ajuste fino (*fine tuning*). No primeiro caso, as camadas transferidas são congeladas, ou seja, os pesos transferidos não são ajustados durante o treinamento na nova tarefa. No segundo caso, todas ou algumas camadas superiores transferidas são descongeladas e os pesos são ajustados na nova tarefa.

### 2.5.1 Estratégias de transferência de aprendizado

A escolha da técnica de transferência de aprendizado depende de diversos fatores, sendo que os principais são: tamanho do conjunto de dados de destino e similaridade com o conjunto de dados original, bem como o número de parâmetros nas primeiras  $n$  camadas [CHOLLET 2016]. Existem quatro estratégias utilizadas para transferência de aprendizado:

- 1) Novo conjunto de dados é pequeno e similar ao conjunto da base original.  
Se o conjunto de dados de destino for pequeno e o número de parâmetros for grande, ajustar todas as camadas pode causar *overfitting*. Neste caso, como o novo conjunto de dados é similar ao original, todas as camadas podem ser transferidas e congeladas, usando a rede-base como um extrator de atributos da tarefa-destino. A extração de atributos consiste em usar as representações aprendidas por uma rede-base para extrair atributos interessantes do novo conjunto de dados. O restante da rede-destino é treinado do zero.

- 2) Novo conjunto de dados é pequeno e muito diferente do original.  
Como o conjunto de imagens é pequeno, mas bem diferente da original, é melhor não utilizar as camadas superiores específicas da rede pré-treinada. Neste caso, a melhor opção é congelar apenas as primeiras camadas da rede-base para extração de atributos e ajustar as camadas superiores. Ajustar todas as camadas com um conjunto de dados pequeno pode causar *overfitting*.
- 3) Novo conjunto de dados é grande e similar ao original.  
Como os conjuntos de imagens são similares, é bem provável que a utilização da CNN pré-treinada, com todas as camadas congeladas, produza bons resultados na nova tarefa. No entanto, como o novo conjunto de imagens é grande, ou, se o número de parâmetros for pequeno, de modo que seja improvável ocorrer problemas de *overfitting*, provavelmente o ajuste de todas as camadas transferidas da rede-base melhoraria o desempenho.
- 4) Novo conjunto de dados é grande e muito diferente do original.  
Neste caso, a melhor opção é o ajuste de todas as camadas transferidas. Embora os conjuntos de imagens sejam bem diferentes, é bem provável que o ajuste de todas as camadas da CNN melhore o desempenho em comparação à inicialização aleatória dos pesos.

Note que nas estratégias 3 e 4, se o conjunto de dados de destino for muito grande, haverá pouca necessidade de transferência.

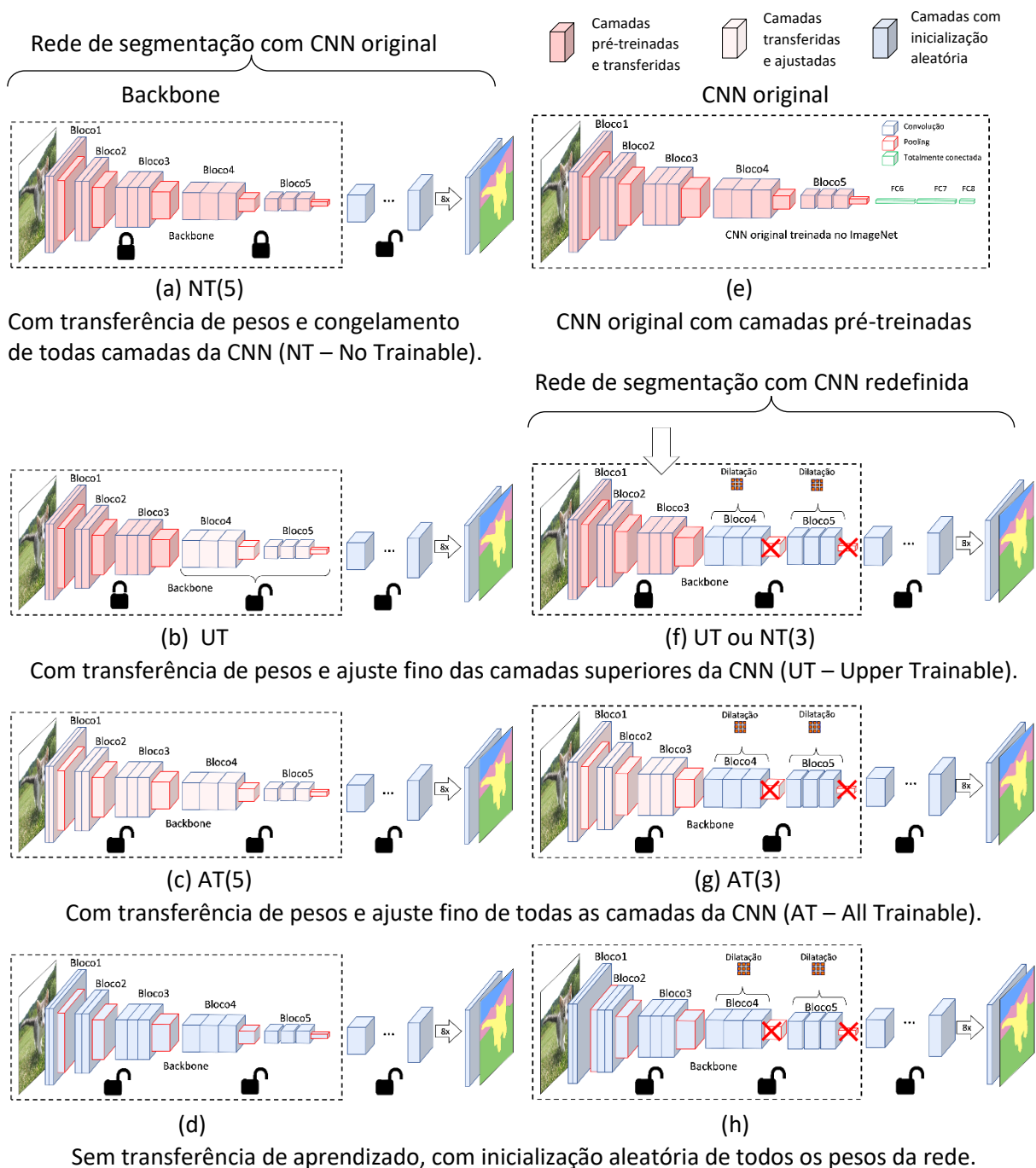
### 2.5.2 Transferência de aprendizado da CNN para redes de segmentação

As redes de segmentação semântica baseadas em CNN utilizam como *backbone* uma CNN de uma tarefa de classificação de imagens, adaptando-a para uma tarefa-destino de segmentação densa de *pixels*. Desta forma, as CNNs pré-treinadas no conjunto ImageNet podem ser transferidas para as redes de segmentação semântica baseadas em CNN, para uma tarefa específica de segmentação de *pixels* do novo conjunto de destino.

A [Figura 2.123](#) (coluna esquerda) representa os modelos de segmentação, tais como FCN, U-Net, SegNet, RefineNet, BiSegNet, que usam *backbones* semelhantes à CNN original ([Figura 2.123e](#)). Assim, todas as camadas da CNN (exceto o classificador FC) são transferidas à rede de segmentação. Na ilustração (a), todos os pesos das cinco camadas são copiados da base pré-treinada e congelados [NT(5) – *No Trainable*]; na ilustração (b), apenas as camadas superiores são ajustadas [UT – *Upper Trainable*]; e na ilustração (c), todas as camadas da rede-base são transferidas e ajustadas [AT(5) – *All Trainable*]. Em todos os casos, as camadas restantes da rede de segmentação são inicializadas aleatoriamente e treinadas no conjunto de dados de destino. Na ilustração (d), não há transferência de aprendizado e todos os pesos são inicializados aleatoriamente.

A [Figura 2.123](#) (coluna direita) representa os modelos de segmentação semântica, tais como PSPNet, DeepLab-v3, DeepLab-v3+ e DenseASPP, que redefinem uma ou duas camadas superiores da CNN, usando convolução *atrous* e removendo operações de subamostragem. Neste caso, não é possível transferir e congelar todas as camadas da rede-base, já que as camadas superiores foram redefinidas. Desta forma, apenas 3 ou 4 camadas da rede-base são transferidas. Na ilustração (e), a CNN original é mostrada com todas as camadas convolucionais (inclusive o classificador FC a ser descartado) pré-treinadas no

ImageNet. Na ilustração (f), as três primeiras camadas são transferidas e congeladas [NT(3) – *No Trainable*]; na ilustração (g), as três camadas transferidas são descongeladas e, portanto, todas as camadas são treinadas [AT(3) – *All Trainable*]; e na ilustração (h), todas as camadas são inicializadas com pesos aleatórios. Em todos os casos, as camadas superiores redefinidas da rede-base e as camadas restantes da rede de segmentação são inicializadas aleatoriamente e treinadas no conjunto de dados de destino.

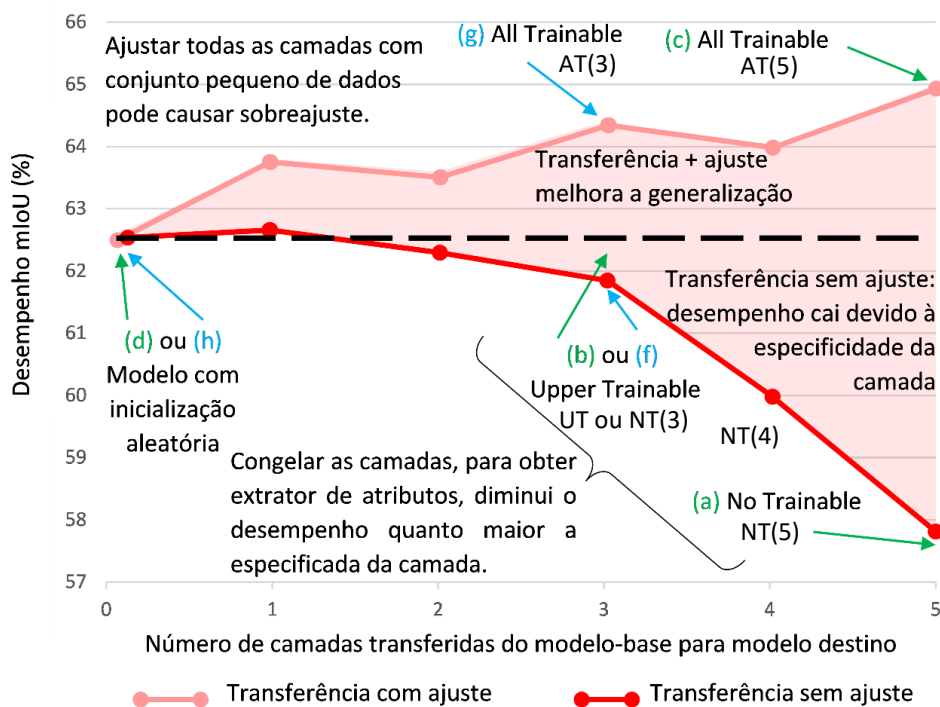


**Figura 2.123** – Configurações dos experimentos com ou sem transferência de pesos para modelos de segmentação semântica com backbone semelhante à CNN original (FCN, U-Net, SegNet, RefineNet, BiSeNet) (esquerda) ou backbone com camadas superiores redefinidas (PSPNet, DeepLab-v3, DeepLab-v3+ e DenseASPP), sendo as camadas pré-treinadas (rosa-escuro), ajustadas (rosa-claro) ou inicializadas com pesos aleatórios (azul).

De acordo com Yosinski *et al.*, a capacidade de transferência de atributos diminui à medida que a diferença entre a tarefa-base e a tarefa-destino aumenta, mas mesmo assim a transferência de atributos com reajuste pode ser melhor do que usar inicialização com atributos aleatórios.

No gráfico de desempenho da Figura 2.124, reinterpretado de [YOSINSKI *et al.* 2014], as camadas 1 e 2 (na linha vermelho-escuro) são transferidas (e congeladas), quase perfeitamente, do modelo pré-treinado ao modelo-destino, evidenciando que, pelo menos para as duas tarefas, os atributos da primeira e segunda camadas são gerais. A transferência sem reajuste de três camadas mostra uma leve queda. Com quatro ou cinco camadas transferidas e congeladas, há uma queda mais significativa no desempenho. Desta forma, os atributos das camadas superiores são transferidos de forma mais fraca (ou seja, são mais específicos) quando os conjuntos de dados de origem e destino são diferentes [YOSINSKI *et al.* 2014].

Tendo isso em mente, nas redes de segmentação semântica baseadas em CNN, usando modelos *backbone* pré-treinados no ImageNet para detecção de plantas daninhas em culturas agrícolas, conforme apresentado neste trabalho, é esperado que as camadas superiores da CNN sejam mais específicas e com menor capacidade de transferibilidade.



Desempenho das ilustrações da Figura 2.123:

(c) AT(5) > (d) Aleatório > (b) UT > (a) NT(5) → (Modelo de 5 camadas transferidas)

(g) AT(3) > (h) Aleatório > (f) NT(3) → (Modelo de 3 camadas transferidas)

**Figura 2.124** – Resultados de desempenho esperados na comparação do modelo com inicialização aleatória e com transferência de aprendizado. Linha tracejada é o desempenho mIoU do modelo de segmentação com inicialização aleatória. Os pontos vermelho-escuros são os desempenhos do modelo de segmentação usando uma CNN pré-treinada, com  $n$  camadas transferidas e congeladas. Os pontos vermelho-claros são as mesmas versões transferidas e ajustadas. Reinterpretado de [YOSINSKI *et al.* 2014].

O gráfico da [Figura 2.124](#) sugere que a inicialização totalmente aleatória do modelo de segmentação, por exemplo, nas ilustrações (d) ou (h), funciona melhor do que usar o modelo com camadas da CNN-base transferidas e congeladas, como nas ilustrações correspondentes (a) NT(5) e (f) NT(3), respectivamente. No entanto, segundo Yosinski *et al.*, a transferência de pesos com reajuste, por exemplo, nas ilustrações (c) AT(5) ou (g) AT(3), resulta em redes que generalizam melhor do que aquelas treinadas diretamente no conjunto de dados de destino com inicialização aleatória, como nas ilustrações correspondentes (d) ou (h). Em outras palavras, inicializar uma rede com transferência de atributos pode melhorar o desempenho de generalização, mesmo após o ajuste no conjunto de dados de destino, aumentando o desempenho da rede neural. Portanto, o efeito de ver o conjunto de dados da tarefa-base persiste mesmo após o reajuste [[YOSINSKI \*et al.\* 2014](#)].

Conseqüentemente, a transferência de pesos seguida de um ajuste de qualquer número de camadas é melhor do que treinar todas as camadas do zero com inicialização aleatória na tarefa-destino. Entretanto, deve-se ter cuidado para evitar problemas de generalização com a ocorrência de sobreajuste da rede (*overfitting*) ao usar um conjunto de dados pequenos.

### 2.5.3 Ajuste fino

Uma etapa de *fine-tuning* consiste em ajustar os pesos das camadas transferidas, treinando o modelo em um novo conjunto de dados para obter melhorias de desempenho. Segundo Chollet, durante o ajuste fino, não é adequado misturar pesos pré-treinados com pesos inicializados aleatoriamente. Em outras palavras, o treinamento deve ser feito em duas etapas. Na primeira, as camadas da CNN-base são transferidas e congeladas, e as camadas restantes do modelo de segmentação são treinadas até a convergência. Depois, as camadas pré-treinadas são descongeladas, e todas as camadas são treinadas conjuntamente na segunda etapa. Se o modelo ainda não estiver treinado, o sinal de erro que se propaga pelas camadas inicializadas aleatoriamente durante o treinamento será grande e as representações previamente aprendidas pelas camadas serão destruídas com grandes atualizações de pesos [[CHOLLET 2016](#)].

Depois que o novo modelo converge (com a configuração NT – *No Trainable*), todas as camadas ou algumas camadas superiores do modelo CNN-base são descongeladas, com objetivo de treinar novamente todo o modelo de segmentação, de ponta a ponta, com uma taxa de aprendizado muito baixa. Desta forma, as camadas superiores mais profundas, que codificam atributos mais especializados do conjunto de dados de base, são ajustadas no novo conjunto. Já as primeiras camadas, que codificam atributos mais genéricos e reutilizáveis para diferentes tarefas, como detectores de bordas, texturas e cores, podem ser preservadas ou ajustadas. Quanto mais camadas são treinadas, maior o risco de *overfitting* com um pequeno conjunto de dados. Assim, uma boa estratégia é ajustar apenas duas ou três camadas superiores do modelo-base por um número menor de épocas de treinamento [[CHOLLET 2016](#)].

Em resumo, o ajuste fino consiste em descongelar todas ou algumas das camadas superiores de uma rede-base pré-treinada e retreinar essas camadas descongeladas junto com as camadas restantes da rede-destino. O ajuste fino ajusta levemente as representações mais abstratas do modelo que está sendo reutilizado, a fim de torná-las mais relevantes ao problema em questão [[CHOLLET 2016](#)].

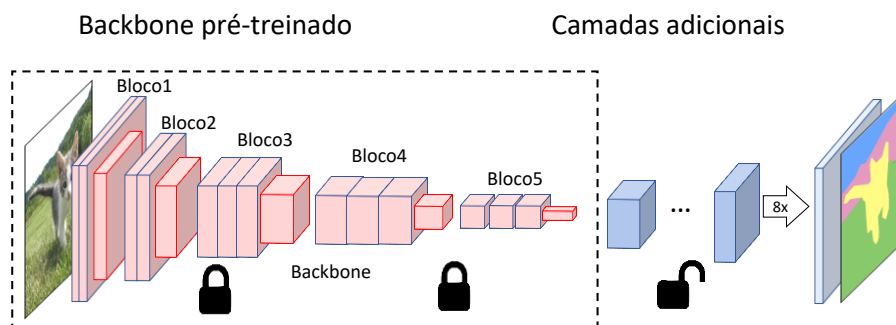
Desta forma, o fluxo de trabalho de aprendizado por transferência com ajuste fino deve ser realizado em duas etapas de treinamento:

Primeira etapa (Figura 2.125a):

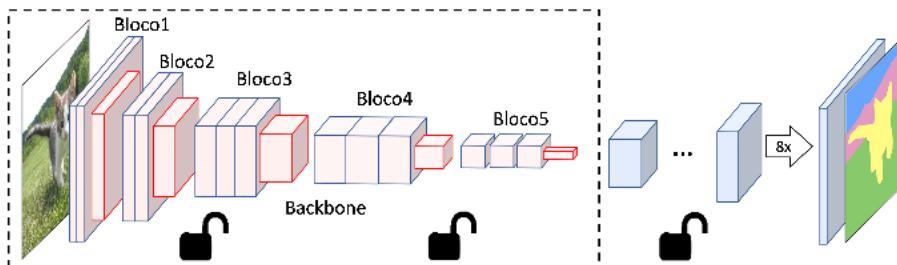
- 1) Instanciar um modelo-base com pesos pré-treinados;
- 2) Congelar todas as camadas pré-treinadas no modelo-base (NT – *No Trainable*, definindo `model_base.trainable = False`), para evitar a destruição de qualquer informação durante as próximas rodadas de treinamento;
- 3) Criar um modelo de segmentação no topo da rede, adicionando algumas camadas novas com inicialização aleatória de pesos treináveis na saída de uma (ou de várias) camadas congeladas do modelo base. Essas camadas aprenderão a transformar os atributos de classificação do conjunto de dados de origem em previsões densas do novo conjunto de dados;
- 4) Treinar as novas camadas do modelo de segmentação no novo conjunto de dados;

Segunda etapa (Figura 2.125b):

- 5) Em seguida, executar o ajuste fino, que consiste em descongelar todo o modelo obtido acima (ou parte dele) e treiná-lo novamente nos novos dados com uma taxa de aprendizado muito baixa. Neste caso, é fundamental usar uma taxa de aprendizado baixa e poucas épocas de treinamento, porque o modelo treinado é muito maior do que na primeira etapa, e o conjunto de dados geralmente é pequeno. Como resultado, há o risco de ocorrer *overfitting* rapidamente, se aplicar grandes atualizações de peso com taxas de aprendizado maiores.



(a) Primeira etapa: camadas pré-treinadas e congeladas (rosa-escuro) para treinamento das camadas adicionais da rede de segmentação, inicializadas aleatoriamente (azul-escuro), no conjunto de dados de destino.



(b) Segunda etapa: treinamento de todo (ou parte) o modelo no conjunto de dados de destino, para ajuste fino das camadas pré-treinadas (rosa-claro) com taxa de aprendizado e número de épocas de treinamento menores.

**Figura 2.125** – Treinamento em duas etapas para ajuste fino de pesos pré-treinados.





## Capítulo 3

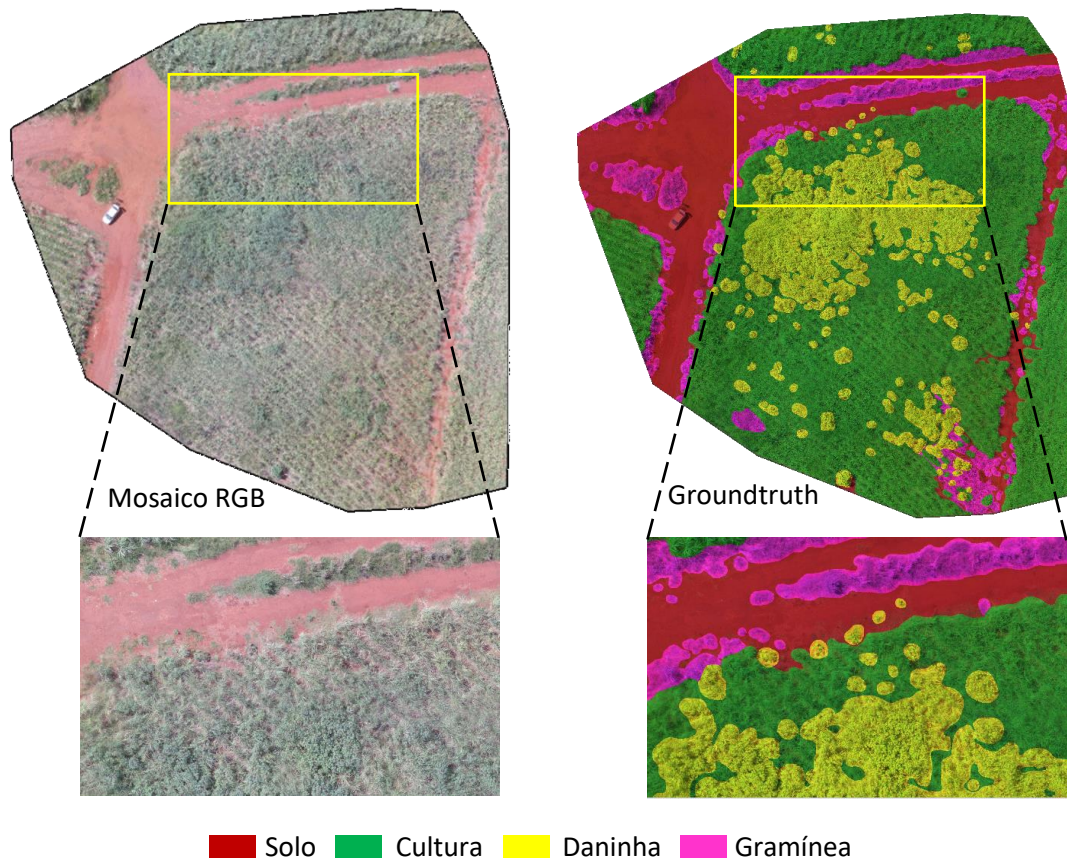
# Trabalhos correlatos de segmentação semântica na agricultura

Com o advento do sensoriamento remoto em plataforma de VANTs (veículos aéreos não tripulados), algumas tecnologias avançadas de visão computacional e inteligência artificial – como os algoritmos de aprendizado de máquina e redes neurais de aprendizado profundo – desempenharam um papel significativo na agricultura de precisão, trazendo melhorias na tarefa de classificação e segmentação de culturas/daninhas. Na literatura, várias abordagens usando aprendizado de máquina convencionais, como KNN (*K-Nearest Neighbors*), máquina de vetores de suporte (*Support Vector Machine* – SVM) [PEREZ-ORTIZ *et al.* 2016; GUERRERO *et al.* 2012, FLETCHER *et al.* 2016], árvores de decisão (*Decision Trees* – DT) [LOTES *et al.* 2017; SANDINO *et al.* 2018] e floresta aleatória (*Random Forest* – RF) [GAO *et al.* 2018], foram desenvolvidas para classificar e detectar plantas daninhas, por meio da extração artesanal de atributos (*feature engineering*) em imagens aéreas de diversas culturas agrícolas capturadas por diferentes sistemas de aquisição embarcados em VANT.

Recentemente, houve uma mudança de paradigma, com abordagens orientadas ao aprendizado supervisionado, usando CNNs para capturar uma representação hierárquica de atributos a partir dos dados de entrada, aumentando o desempenho de muitas tarefas, incluindo classificação de imagens, detecção de objetos e segmentação semântica [César Pereira *et al.* 2019]. Por exemplo, Huang *et al.* compararam as redes FCN baseadas em diferentes CNNs pré-treinadas no ImageNet (AlexNet, VGG-16 e GoogLeNet) para segmentação de plantas daninhas em imagens aéreas de VANT, capturadas a seis metros de altura sobre um campo de arroz, obtendo melhor desempenho com a rede FCN-8s/VGG-16 [HUANG *et al.* 2018]. Além disso, neste capítulo, revisamos três trabalhos de pesquisa que usam redes CNN para classificação [FERREIRA *et al.* 2017] ou redes de segmentação semântica baseadas em CNN [SA *et al.* 2017; SA *et al.* 2018] para detecção de plantas daninhas a partir de imagens de VANTs.

No domínio da agricultura, onde os limites dos objetos não são bem-definidos, as redes de segmentação semântica a nível de *pixel* se mostraram mais adequadas para identificação de plantas daninhas do que os modelos de CNN para classificação de imagens, detecção de objetos ou segmentação de instâncias. Em outras palavras, as redes de classificação, que geram um único rótulo para uma imagem, não são adequadas para classificação densa a nível de *pixel* para detectar plantas daninhas em blocos de grandes imagens de alta resolução espacial. Além disso, agrupamentos de plantas podem ser de difícil delimitação, principalmente em redes de detecção de objetos que preveem a classe e

a caixa delimitadora de cada objeto (*bounding box*). Portanto, ferramentas mais refinadas usando *zoom in/out* são utilizadas para anotação a nível de *pixel* em classes semânticas de objetos (por exemplo, daninha, cultura, solo, etc.), em vez de anotação de rótulo único por bloco de imagem ou anotação de contorno baseada em polígonos. Já a anotação a nível de *pixel* de instâncias de plantas daninhas seria impraticável para segmentação de instâncias, que separam os *pixels* de todas as instâncias de cada objeto. Na [Figura 3.1](#), mostramos um exemplo de um mosaico de imagens aéreas utilizado neste trabalho, com anotação semântica de *pixels* em quatro classes (solo, cultura, daninha e gramínea).



**Figura 3.1** – Mosaico de imagens aéreas com anotação de *pixels* em quatro classes.

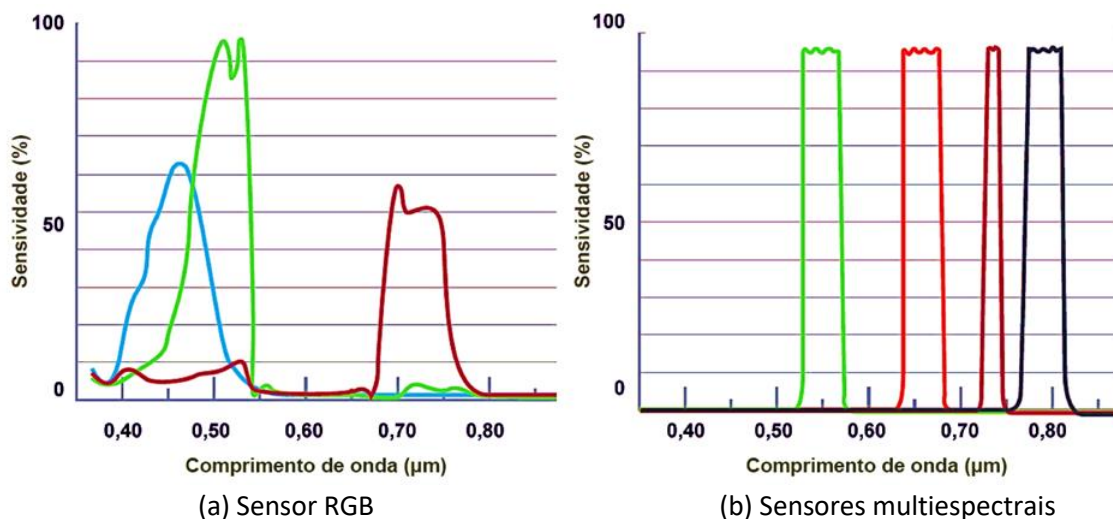
Muitos estudos sobre segmentação semântica de culturas/daninhas consideram apenas imagens individuais capturadas durante o mapeamento aéreo com VANT, que cobrem apenas algumas dezenas de metros quadrados [[HUANG et al. 2018](#); [FERREIRA et al. 2017](#); [SA et al. 2017](#)]. No entanto, a maioria das aplicações práticas requer mapas que cobrem grandes áreas (da ordem de dezenas de hectares), preservando os detalhes das distribuições de plantas. Para abordar este problema, alguns trabalhos utilizam como entrada mapas ortomosaicos georreferenciados que representam as propriedades de toda a área de cultivo em escala métrica ([Figura 3.1](#)). Deste modo, com um mapeamento aéreo de alta resolução, é possível monitorar campos agrícolas para aplicação seletiva e automatizada de agroquímicos em direção a uma agricultura sustentável. Sendo assim, apresentamos a revisão de um trabalho que usa uma rede neural profunda para segmentação de culturas/daninhas a partir de ortomosaicos de imagens multiespectrais de VANT [[SA et al. 2018](#)]. Como as imagens aéreas de alta resolução de VANT permitem capturar detalhes espaciais de culturas e daninhas, é possível obter uma extração de atributos úteis e discriminativos, aumentando o desempenho do sistema.

A maioria dos estudos na área de sensoriamento remoto e, particularmente, na agricultura de precisão usa imagens multiespectrais e mapas de índices de vegetação. Por isso, iniciamos com uma introdução sobre este assunto, destacando um trabalho de Sanchez *et. al.* [TORREZ-SÁNCHEZ *et al.* 2013] com um estudo dos índices de vegetação aplicados em classificação de daninhas/culturas em imagens de VANT.

Nesta tese, levamos em consideração os problemas e desafios levantados por estes trabalhos correlatos e propomos uma solução para melhorar o desempenho de segmentação de plantas daninhas em ortomosaicos de imagens aéreas de VANT.

### 3.1 Imagens multiespectrais e mapas de índices de vegetação

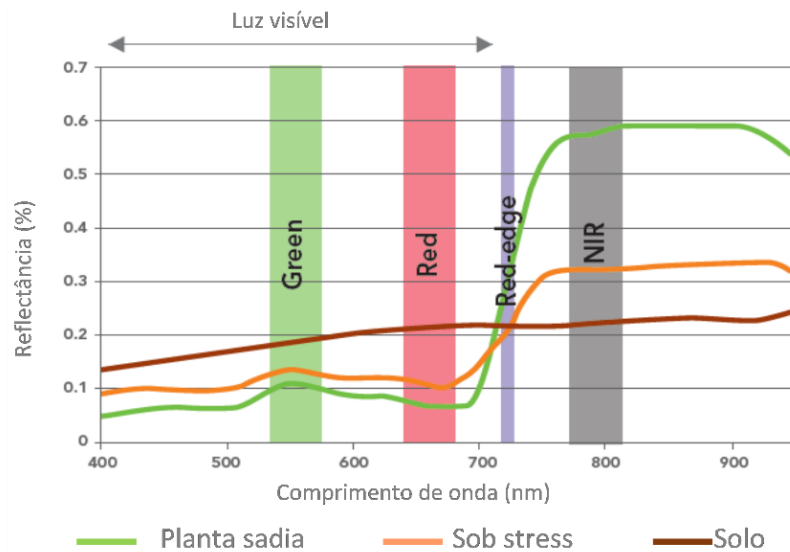
Os VANTs podem ser equipados com uma câmera com sensores RGB e/ou multiespectrais. Os sensores RGB capturam três bandas espectrais com sobreposição entre faixas de comprimento de onda (Figura 3.2a), enquanto os sensores multiespectrais possuem bandas espectrais estreitas com resposta uniforme sem sobreposição. A Figura 3.2b mostra um exemplo de espectros das bandas de um sensor multiespectral: *Green* (comprimento de onda 550nm, 40nm de largura de banda), *Red* (660nm, 40nm de largura), *Red-Edge* (735nm, 10nm de largura mais estreita) e *Near Infrared* (790nm, 40nm de largura). Desta forma, as bandas multiespectrais fornecem valores de reflectância – quociente da energia refletida (radiância) pela energia incidente (irradiância) no mesmo instante de tempo – mais confiáveis para classificação de objetos em sensoriamento remoto, por meio do comportamento espectral de alvos terrestres [HOLLER *et. al.* 2022].



**Figura 3.2** – Espectro de ondas eletromagnéticas das bandas espectrais do sensor RGB (a); espectro de quatro bandas dos sensores multiespectrais (b) [HOLLER *et. al.* 2022].

A Figura 3.3 mostra que o comportamento espectral de uma planta saudável possui reflectância no espectro visível, principalmente na faixa de luz verde, como mostra a curva de reflectância (linha verde). Por outro lado, a clorofila absorve a energia solar ocasionando uma baixa reflectância na faixa do vermelho. No entanto, as plantas saudáveis têm uma reflectância ainda maior na faixa do infravermelho próximo (NIR) e na borda do vermelho

(*Red-Edge*), invisíveis ao olho humano. Portanto, a vegetação sadia pode ser identificada pela alta reflectância no NIR e baixa reflectância no espectro visível [HOLLER *et. al.* 2022].



**Figura 3.3** – *Comportamento espectral da vegetação e do solo* [HOLLER *et. al.* 2022].

As propriedades da refletância da vegetação podem ser melhor avaliadas por meio de combinações matemáticas de diferentes bandas espectrais, denominadas índices de vegetação. Alguns índices são calculados com as bandas espectrais visíveis, tais como:

$$\text{ExG} = 2G - R - B,$$

$$\text{NGRDI} = (G - R) / (G + R),$$

$$\text{VARI} = (G - R) / (G + R - B),$$

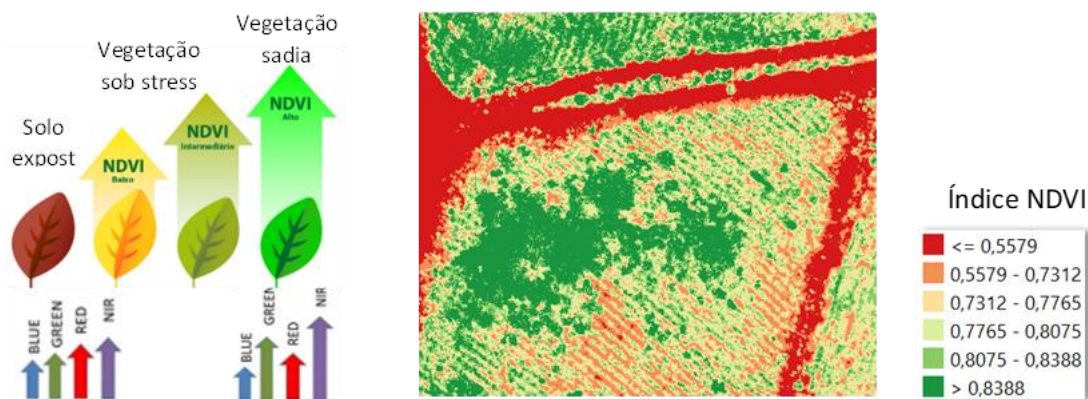
onde o índice ExG (*Excess Green Index*) é calculado nos canais RGB destacando o canal verde, enquanto o índice NGRDI (*Normalized Green-Red Difference Index*) é calculado pela diferença normalizada dos canais *Green* e *Red*. O índice VARI (*Visible Atmospherically Resistant Index*) é baseado na variabilidade de vigor e estresse da planta (pela quantidade de verde) [JENSEN 2009].

O índice NDVI (*Normalized Difference Vegetation Index*) [ROUSE *et al.* 1974] é um dos mais utilizados para estudo da vegetação e monitoramento agrícola, calculado pela diferença entre as reflectâncias das bandas NIR e *Red*, dividido pela soma das reflectâncias dessas bandas, ou seja,

$$\text{NDVI} = (\text{NIR} - R) / (\text{NIR} + R).$$

A relação entre as bandas NIR e *Red* indica a condição da vegetação devido à absorção de clorofila dentro da faixa espectral vermelha e alta reflectância dentro da faixa NIR. Portanto, quanto maior o contraste, mais sadia a vegetação. Sendo assim, o resultado do índice NDVI varia de -1 a 1, de modo que quanto mais próximo de 1, maior o indício de presença de vegetação saudável, e quanto mais próximo de -1, maior o indício de vegetação morta ou presença de solo exposto, como mostra o mapa NDVI colorido na Figura 3.4.





**Figura 3.4** – Mapa de índice de vegetação por diferença normalizada (NDVI).

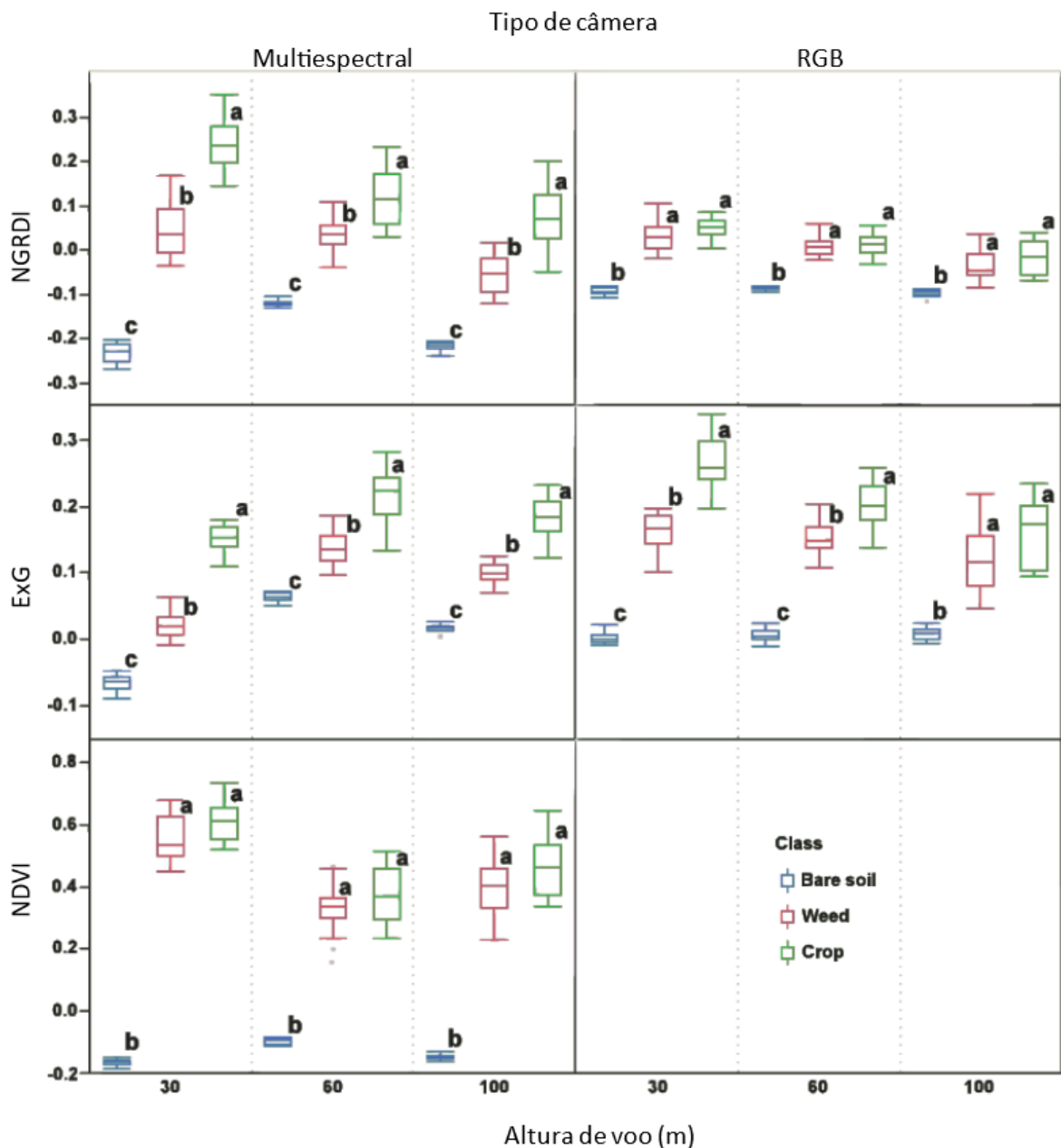
A Tabela 3.1 lista algumas das centenas de índices existentes na literatura [JENSEN 2009]. Estes índices de vegetação são calculados usando diferentes bandas espectrais, sendo que cada índice tem uma finalidade diferente. Desta forma, as imagens RGB e multiespectrais (NIR, Red, Green, Blue, Red-Edge) têm permitido criar mapas de índices de vegetação que são associados a diversas propriedades fisiológicas da vegetação, como saúde, vigor e estresse hídrico. Além disso, diferentes formações vegetais apresentam comportamento distinto em termos de resposta espectral. Por este motivo, câmeras multiespectrais são bastante usadas na agricultura [HOLLER et al. 2022; SHIRATSUCHI et al. 2014; JENSEN 2009].

**Tabela 3.1** – Alguns exemplos de índices de vegetação.

Índice	Descrição	Fórmula	Tipo de imagem
SR	Simple Ratio [Birth 1968]	NIR/Red	Multiespectral
GNDVI	Green Normalized Difference Vegetation Index [GITELSON1996]	$(NIR - Green) / (NIR + Green)$	Multiespectral
LCI	Leaf Chlorophyll Index	$(NIR - RedEdge) / (NIR + Red)$	
NDRE	Normalized Difference Red Edge [BARNES et al. 2000]	$(NIR - RedEdge) / (NIR + RedEdge)$	
NDVI	Normalized Difference Vegetation Index [ROUSE 1974]	$(NIR - Red) / (NIR + Red)$	
SIPI2	Structure Intensive Pigment Index 2	$(NIR - Green) / (NIR - Red)$	
MCARI	Modified Chlorophyll Absorption in Reflective Index [HABOUDANE et al. 2004]	$1.2 * (2.5 * (NIR - Red) - 1.3 * (NIR - Green)) / (\text{normalizado com valor máximo de Red, Green e NIR})$	
NGRDI	Normalized Green-Red Difference Index	$NGRDI = (G - R) / (G + R)$	RGB
TGI	Triangular Greenness Index	$(G - (0.39 * R) - (0.61 * B)) / (\text{normalizado com valor máximo R, G e B})$	ou
GLI	Green Leaf Index	$((2 * G) - R - B) / ((2 * G) + R + B)$	Multiespectral (Red, Green, Blue)
ExG	Excess Green Index	$(2 * G) - (R + B)$	
VARI	Visible Atmospherically Resistant Index	$\min(1, \max(-1, (G - R) / (G + R - B)))$	

Nota:  $\min(1, \max(-1, ...))$  fixa os valores entre -1 e 1.

Sanchez *et. al.* [TORREZ-SÁNCHEZ *et al.* 2013] mostraram que os valores dos índices de vegetação são afetados pela altura do voo e tipo de câmera, como mostra a **Figura 3.5**. Diferenças espectrais entre vegetação (plantas daninhas + cultura) e solo exposto foram significativos para todos os índices, principalmente a uma altura de 30 metros, sendo que o índice NDVI alcançou a maior separabilidade espectral entre essas duas classes, sugerindo o emprego de imagens multiespectrais para uma discriminação mais robusta. Isto pode ser explicado devido ao fato de o NDVI enfatizar a resposta espectral da banda NIR, que caracteriza o vigor da vegetação, e ser menos sensível aos efeitos do solo do que os outros dois índices NGRDI e ExG. No entanto, estes dois últimos índices são relevantes devido ao custo da câmera RGB ser menor do que da câmera multiespectral.



**Figura 3.5** – Valores de índices de vegetação de cada classe de cobertura de solo (solo exposto, plantas daninhas e cultura). Os valores dos índices são afetados pela altura de voo e tipo de câmera [TORREZ-SÁNCHEZ *et al.* 2013].

A câmera multiespectral mostra capacidade maior para discriminar entre a cultura e as plantas daninhas do que a câmera RGB. Na câmera multiespectral, os índices NGRDI e ExG foram significativamente diferentes para plantas daninhas e cultura em todas as alturas de voo. O melhor desempenho da câmera multiespectral pode ser causado pela largura de banda mais estreita do sensor ([Figura 3.2b](#)). Cada sensor da câmera multiespectral usa filtros com 10 nm a 40 nm de largura de banda, que reduz as interferências causadas por outros comprimentos de onda, enquanto a câmera RGB adquire informações em três bandas mais amplas de onda espectrais em todo o espectro visível ([Figura 3.2a](#)). No entanto, embora o NDVI tenha sido o único índice estudado usando a banda NIR, este não foi capaz de discriminar entre cultura e ervas daninhas nos estágios iniciais de desenvolvimento em qualquer altura de voo ([Figura 3.5](#)).

Com o objetivo de reduzir os requisitos computacionais para geração do mosaico, o ideal seria capturar imagens na maior altura de voo possível, minimizando o número de imagens capturadas. No entanto, diferenças espectrais mais altas foram obtidas nas alturas mais baixas, isto é, os métodos baseados em *pixels* podem não discriminar ervas daninhas e culturas em alturas acima de 30 metros devido à semelhança espectral entre estas classes de vegetação. Portanto, para superar estas limitações de classificação baseada apenas na resposta espectral dos alvos, usando imagens de VANTs em alturas de voo mais elevadas, quando as ervas daninhas e a cultura estão nos primeiros estágios de crescimento, são necessários métodos mais avançados de aprendizado profundo para distinguir e mapear ervas daninhas e culturas [[TORREZ-SÁNCHEZ et al. 2013](#)].

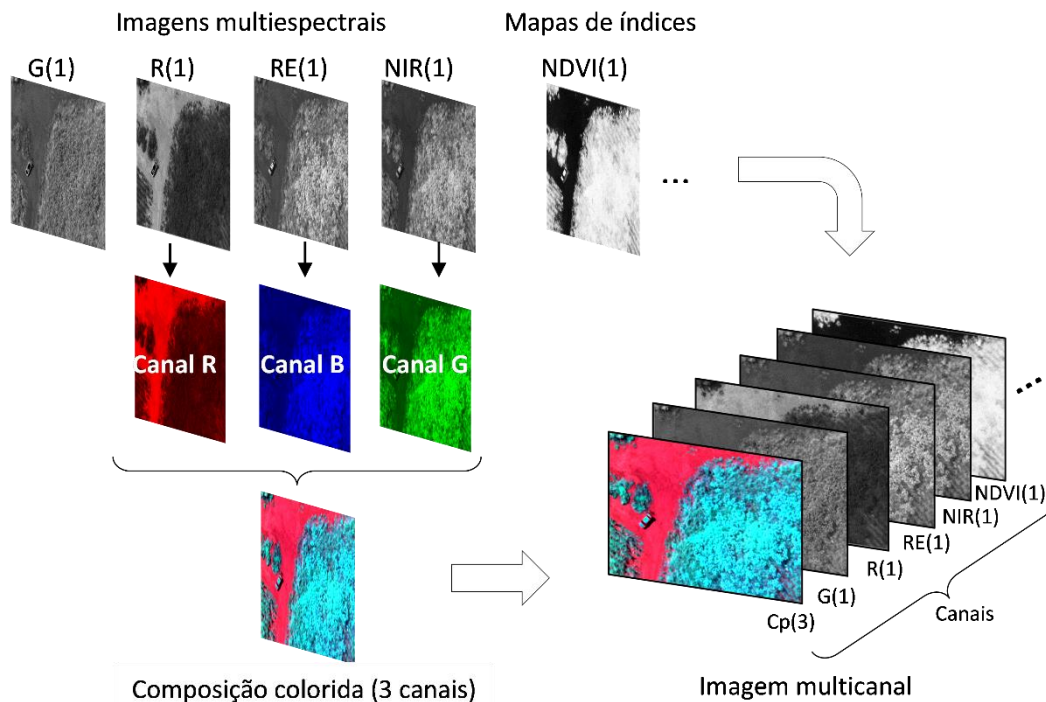
Para aprimorar o desempenho em redes de segmentação semântica, vários estudos adotam diferentes estratégias de entrada das CNNs, usando imagens multicanais com diferentes quantidades de bandas espectrais, dependendo da câmera disponível, como exemplificado na [Tabela 3.2](#). Por exemplo, com câmeras RGB, as imagens de entrada podem ser compostas por 1 a 3 canais (R, G e B), ou mais, empilhando-se mapas de índices baseados em espectros visíveis, tal como NGRDI, VARI e EXG. Com câmeras com sensores RGB e/ou multiespectrais é possível gerar imagens multicanais, empilhando vários canais espectrais e mapas de índices de vegetação.

Câmeras com sensores multiespectrais (*Red*, *Green*, *Blue*, *Red-Edge* e NIR), que capturam imagens com um canal composto de uma banda espectral cada, permitem gerar mapas de índices de vegetação (1 canal), imagens RGB (3 canais: *Red*, *Green* e *Blue*) e imagens de infravermelho colorido (CIR – *Color Infrared*), empilhando os canais NIR, *Red* e *Green*. Assim como as imagens CIR, várias outras composições coloridas, chamadas de composição falsa-cor, por reproduzirem cenas em cores normalmente não vistas pelo olho humano, podem ser geradas pela combinação de diferentes bandas espectrais nos canais RGB, usando o processo aditivo de combinação das cores primárias ([Figura 3.6](#)).

**Tabela 3.2** – Combinação de bandas para diferentes estratégias de classificação.

Tipo de câmera	# de bandas	Combinações de bandas
RGB	3 ou 4	R + G + B + (Índice)
Multiespectral	4 ou 5	Green + Red + RedEdge + NIR + (Índice)
	5 ou 6	Blue + Green + Red + RedEdge + NIR + (Índice)
RGB + Multiespectral	7 ou 8	(R + G + B) + (Green + Red + RedEdge + NIR) + (Índice)
	8 ou 9	(RGB) + (Blue + Green + Red + RedEdge + NIR) + (Índice)





**Figura 3.6** – Exemplos de composição colorida e imagem multicanal a partir de imagens multiespectrais e mapas de índice de vegetação. Número de canais em parênteses.

Nas próximas seções, são apresentadas as revisões de alguns trabalhos que usam redes de segmentação semântica com entrada composta por imagens (ou mosaicos) RGB e/ou multiespectrais de VANTs, com diferentes configurações de imagens multicanais, para detecção de daninhas em diferentes culturas. Com base nos problemas levantados nestes estudos, esperamos desenvolver um método mais avançado usando redes neurais convolucionais para segmentação semântica, que possa apresentar melhor desempenho.

### 3.2 Weed detection in soybean crops using convnets

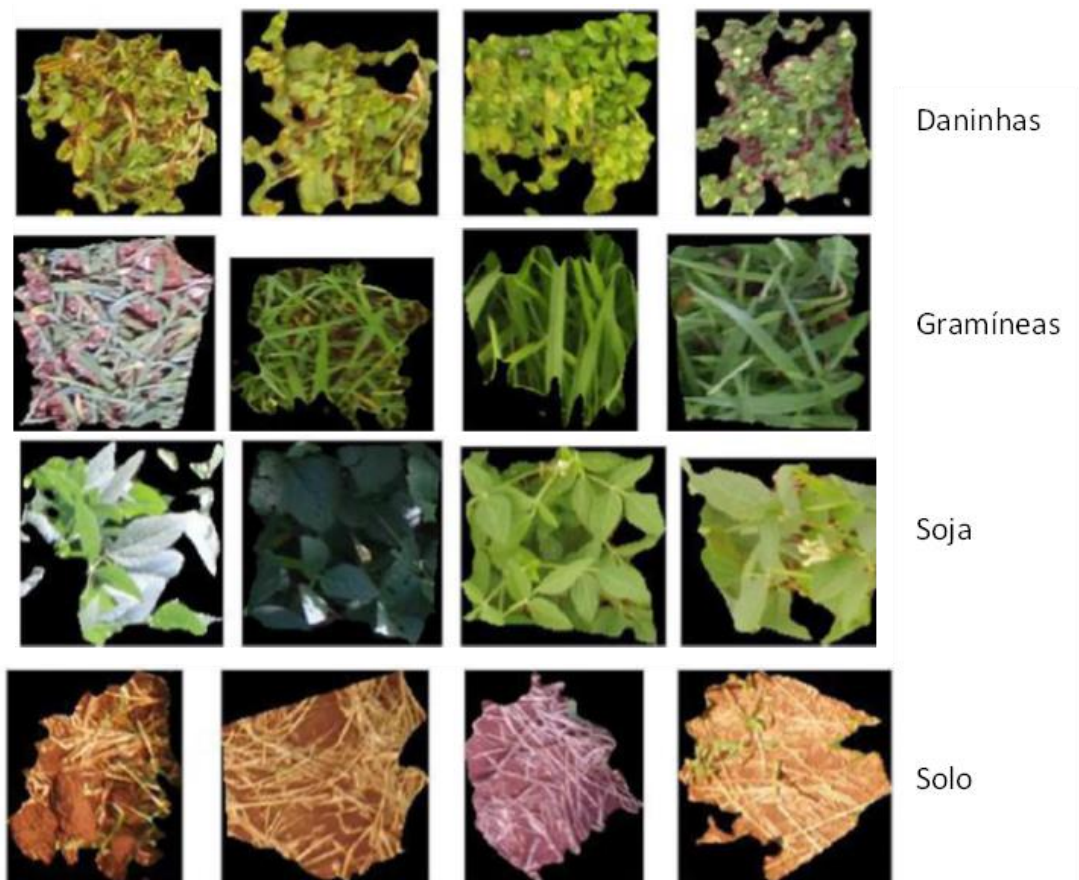
Ferreira *et al.* [FERREIRA *et al.* 2017] usaram redes neurais convolucionais (CNNs) para classificação de plantas daninhas (gramíneas e folhas largas) em imagens de cultura de soja. A abordagem proposta para detecção de plantas daninhas é composta por algumas etapas descritas a seguir.

A primeira etapa consiste na captura de imagens de lavouras de soja, usando um VANT equipado com uma câmera RGB montada em um estabilizador (*gimbal* de 3 eixos), para garantir que as imagens sejam adquiridas em posição estável com o eixo ótico da câmera apontado verticalmente para baixo (posição nadir). As imagens foram capturadas em modo de voo manual a uma altura média de 4 metros acima do nível do solo, que corresponde a uma resolução espacial – GSD (*Ground Sample Distance*) – inferior a 1 cm/pixel. O segundo estágio consiste na segmentação dessas imagens usando o algoritmo de *superpixels* (SLIC) [ACHANTA *et al.* 2012], cujos segmentos extraídos foram anotados manualmente, como mostra a Figura 3.7 e Figura 3.8. Estes segmentos anotados são usados na construção de um conjunto de dados contendo 3.249 segmentos de solo, 7.376 de soja, 3.520 gramíneas e 1.191 de invasoras de folhas largas.



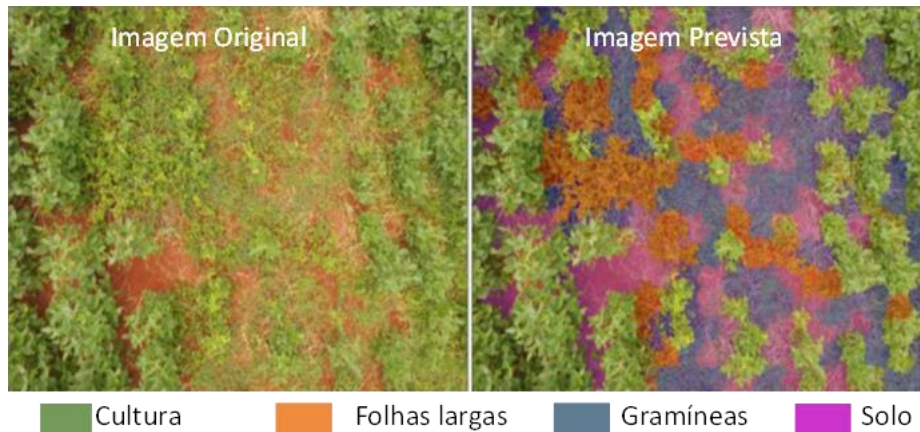
**Figura 3.7** – Imagem parcial segmentada usando o algoritmo SLIC [FERREIRA *et al.* 2017].

A terceira etapa consiste no treinamento de uma CNN usando os segmentos das imagens rotuladas do conjunto de dados (Figura 3.8). A última fase compreende a segmentação e classificação dos segmentos de uma imagem de um plantio de soja quanto à presença de plantas daninhas, usando uma arquitetura de rede neural convolucional baseada na rede AlexNet [KRIZHEVSKY *et al.* 2012].



**Figura 3.8** – Exemplos de classes do conjunto de dados de imagens segmentadas. Em cada linha são mostradas quatro amostras de cada classe: ervas daninhas de folhas largas; gramíneas; soja e solo [FERREIRA *et al.* 2017].

Como resultado, este trabalho alcançou acurácia acima de 98% na classificação de plantas daninhas de folhas largas e gramíneas em relação ao solo e à soja [FERREIRA *et al.* 2017], como mostrado no exemplo da Figura 3.9. Embora seja provável que uma taxa de precisão tão alta não possa ser alcançada em uma aplicação real, uma vez que toda a pesquisa foi realizada em um ambiente controlado, uma taxa de precisão próxima poderia ser alcançada na prática, usando um conjunto de dados de imagem mais diversificado, com tipos variados de solo e ervas daninhas, resoluções de imagens diferentes, etc.



**Figura 3.9** – Imagem original e imagem classificada pela CNN. A cultura de soja foi mantida em sua cor original (verde), laranja representa as ervas daninhas de folhas largas, azul, as gramíneas, e magenta, o solo [FERREIRA *et al.* 2017].

### 3.3 WeedNet: Dense semantic weed classification using multispectral images and MAV for smart farming

Sa *et al.* apresentaram uma abordagem para segmentação semântica de plantas daninhas em um campo experimental de beterraba sacarina usando imagens multiespectrais coletadas por VANT [SA *et al.* 2017]. Neste estudo, foi aplicado o modelo SegNet [BADRINARAYANAN *et al.* 2017] de segmentação semântica baseado em CNN, que extrai informações diretamente das imagens de entrada para classificar diferentes tipos de culturas e ervas daninhas. Uma das contribuições deste sistema foi um estudo sobre a classificação de culturas/ervas daninhas, usando a segmentação semântica densa em imagens multicanais com diferentes canais de entrada multiespectrais.

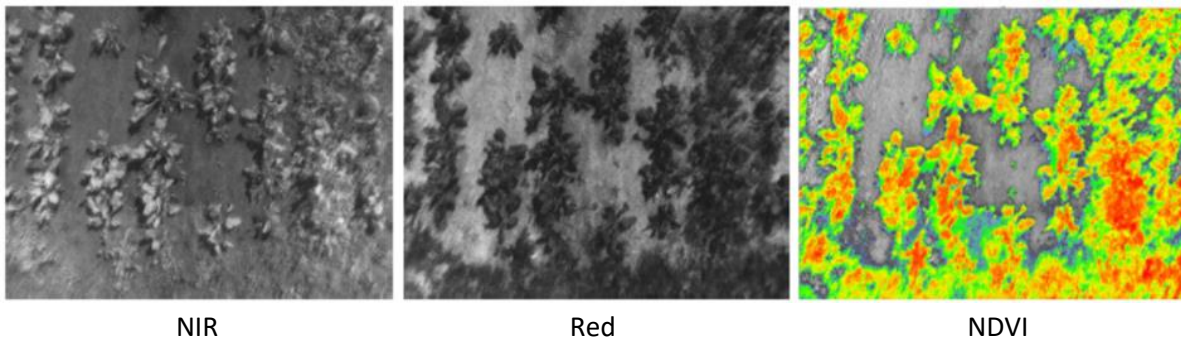
#### 3.3.1 Aquisição do conjunto de dados

Para aquisição de imagens de um campo de beterraba sacarina controlado por herbicidas, com diferentes coberturas de cultura, planta daninha e solo, foi utilizado um veículo aéreo com uma câmera multiespectral Micasense Sequoia. Esta câmera apresenta um sensor RGB e quatro sensores de banda estreita, que capturam imagens monocromáticas multiespectrais nas bandas NIR (*Near Infrared*), *Red*, *Red-Edge* e *Green*, mas apenas as imagens NIR e *Red* foram usadas devido à dificuldade de registro (alinhamento) de imagens de outras bandas [SA *et al.* 2017].

As imagens multiespectrais oferecem a possibilidade de criar índices específicos de vegetação que são amplamente explorados na agricultura. O mapa de índice de vegetação

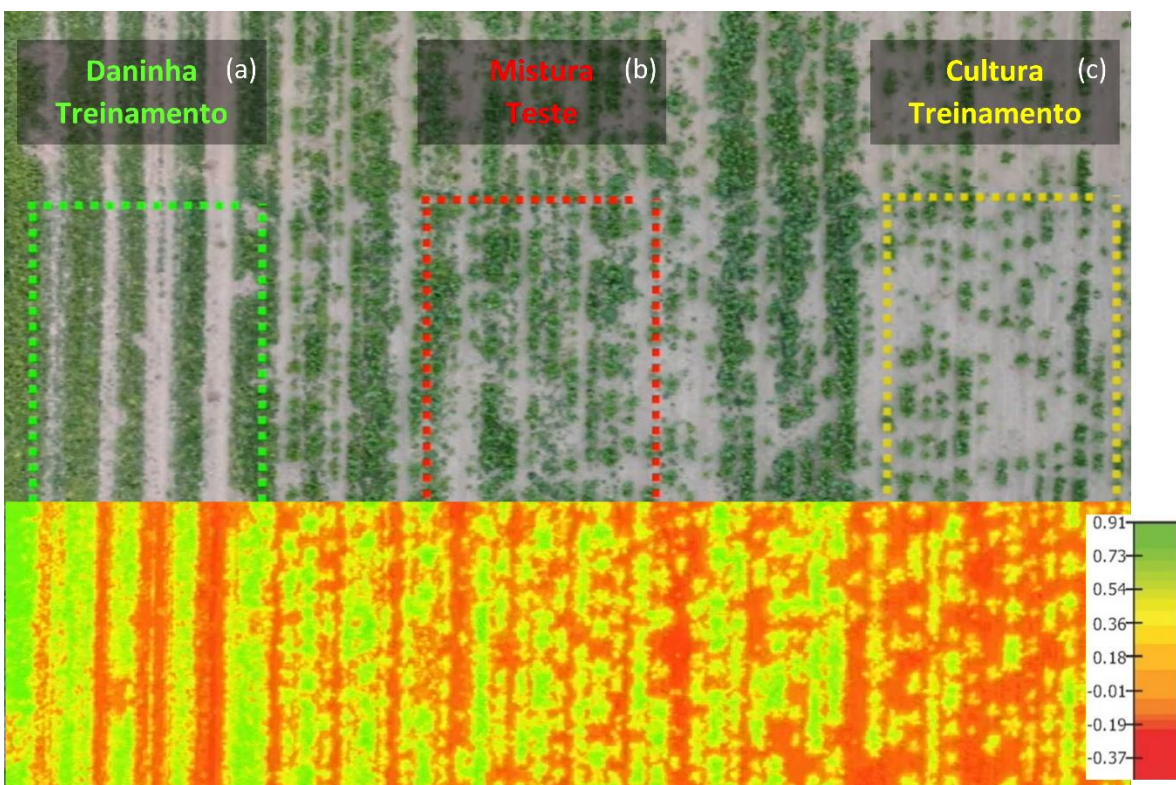


NDVI, calculado a partir das bandas NIR e *Red* (após o alinhamento das imagens por transformação afim), permite discriminar entre o solo e vegetação, por meio de um detector de limiar automático [OTSU 1979] a partir do histograma do mapa NDVI (Figura 3.10).



**Figura 3.10** – Detecção de limite do NDVI extraído das imagens NIR e Red [SA et al. 2017].

Foram coletados três tipos de conjuntos de dados contendo: (a) apenas daninha; (b) tanto cultura como erva daninha; e (c) apenas cultura, em um campo de cultivo de  $40 \times 40$  metros com diferentes níveis de aplicação de herbicidas, como mostra a Figura 3.11. No caso de (a) e (c), é possível criar o *groundtruth* de daninha ou cultura, extraíndo o mapa de índice NDVI, que separa a cobertura vegetal do solo (índices maiores, isto é, cores mais verdes, indicam maior reflectância das plantas saudáveis). No caso de (b), a rotulagem manual foi realizada para discriminar entre daninha e cultura, não diferenciada pelo NDVI. As imagens de (a) e (c) foram usadas para treinamento e (b), para teste.

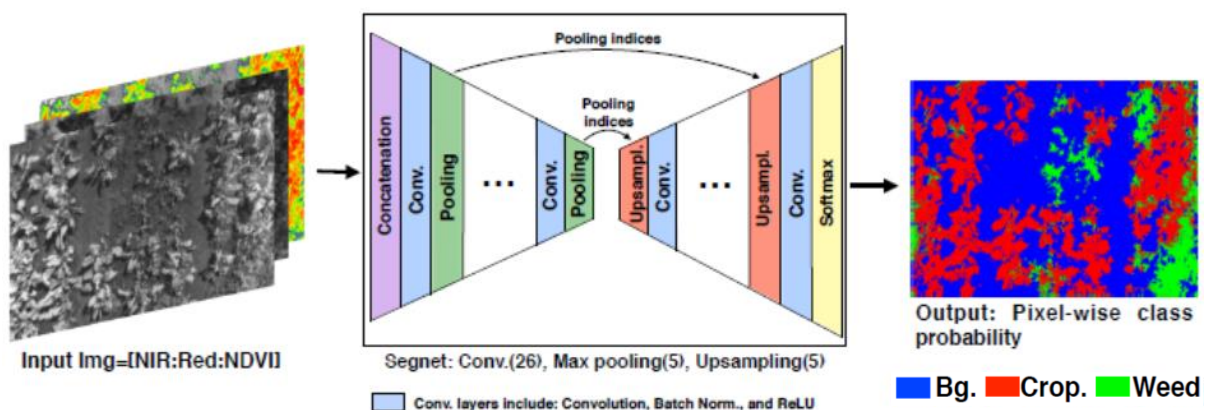


**Figura 3.11** – Vista aérea parcial do campo controlado com níveis variados de herbicida. Uma quantidade máxima de herbicida foi aplicada nas imagens de cultura à direita (retângulo amarelo) e nenhum herbicida foi utilizado nas imagens de ervas daninhas (verde). O centro mostra uma mistura de plantas daninhas e cultura devido ao uso médio de herbicida (vermelho). Mapa de índice NDVI parcial sobre a imagem [SA et al. 2017].

A anotação manual de *pixels* de cultura e ervas daninhas é uma tarefa desafiadora, devido a ambiguidades entre as duas classes, exigindo experiência e conhecimento de um domínio específico. Além disso, são necessárias ferramentas de seleção mais refinadas (por exemplo, seleção no nível de *pixel* e *zoom in/out*), em vez de anotação de contorno baseada em polígonos. Além das imagens de treinamento anotadas automaticamente, foram anotadas manualmente 30 imagens de teste, resultando em um conjunto de dados de 132, 243 e 90 imagens (NIR + *Red* + NDVI) de cultura, ervas daninhas e mistura de plantas daninhas e cultura, respectivamente. As imagens de treinamento/teste multicanais são compostas de imagens multiespectrais alinhadas (NIR, *Red* e o mapa de índice NDVI).

### 3.3.2 Arquitetura de segmentação semântica densa

Uma arquitetura codificador-decodificador, SegNet-v2 [BADRINARAYANAN *et al.* 2017], descrita na Seção 2.4.5, foi utilizada para segmentação densa. Esta rede utiliza uma CNN modificada de classificação de imagens – VGG-16 [SIMONYAN e ZISSERMAN 2015] sem as camadas totalmente conectadas – como modelo-base do codificador. Na Figura 3.12, as imagens multicanais rotuladas são alimentadas na entrada da rede, sendo que uma primeira camada foi inserida para concatenação de qualquer número de canais de entrada, que varia de 1 a 3 canais. A saída corresponde à probabilidade de classes de cada *pixel* – solo, cultura e daninha. A classe *Bg* (azul) indica o fundo (principalmente solo), *Crop* (vermelho) indica beterraba sacarina e *Weed* (verde), plantas daninhas [SA *et al.* 2017].

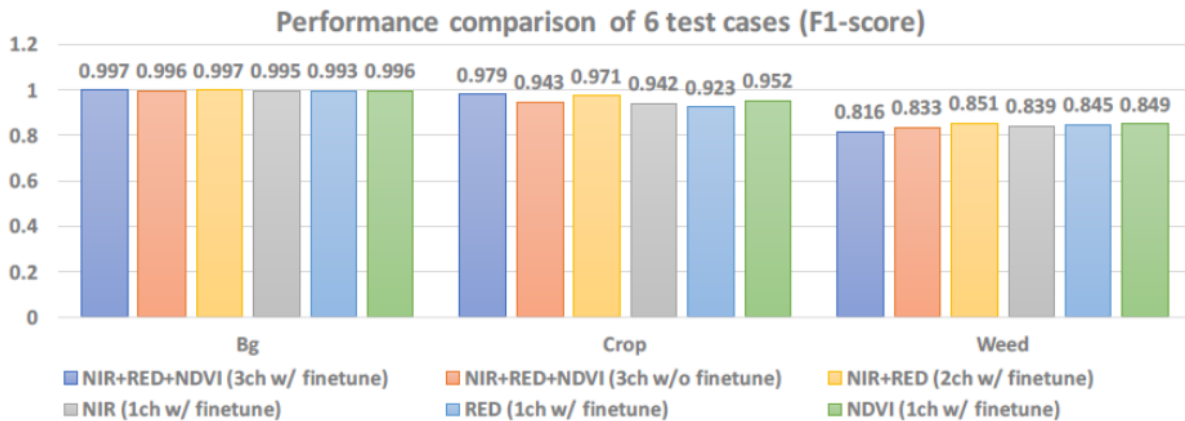


**Figura 3.12** – Arquitetura decodificador-decodificador de segmentação semântica densa [SA *et al.* 2017].

### 3.3.3 Resultados quantitativos

A Figura 3.13 mostra os resultados quantitativos de seis modelos diferentes, treinados com duas condições variáveis (número de canais de entrada e ajuste fino), para ver o impacto dessas variações. Assim, os modelos tem: (a) inicialização aleatória e sem ajuste fino (*w/o finetune*), ou inicialização com pesos da VGG-16 pré-treinada e com ajuste fino (*w/ finetune*); e (b) número de canais de entrada variando de 1 a 3. Todos os modelos apresentam um desempenho de pontuação F1 (*F1-Score*), definido no Apêndice C, acima de 80% para todas as classes, considerando a dificuldade do conjunto de dados [SA *et al.* 2017].

Como mostrado na [Figura 3.13](#), as barras azul-escuras e laranjas (com e sem ajuste fino da última camada, respectivamente), com imagens de entrada de três canais, mostraram que o ajuste fino não afeta significativamente a saída. Isso ocorre, principalmente, em razão dos dois conjuntos de dados de treinamento (beterraba sacarina e ervas daninhas) terem aparência e espectro diferentes (660-790nm) do conjunto de dados ImageNet formado por imagens RGB naturais [[DENG et al. 2009](#)]. Portanto, todos os pesos transferidos devem ser corrigidos durante o treinamento para diminuir o erro de previsão. Entretanto, o tamanho do conjunto de dados é relativamente pequeno em comparação ao ImageNet, que tem 1,2 milhões de imagens e, por essa razão, ajustar todas as camadas com um conjunto de dados pequeno pode causar *overfitting*.



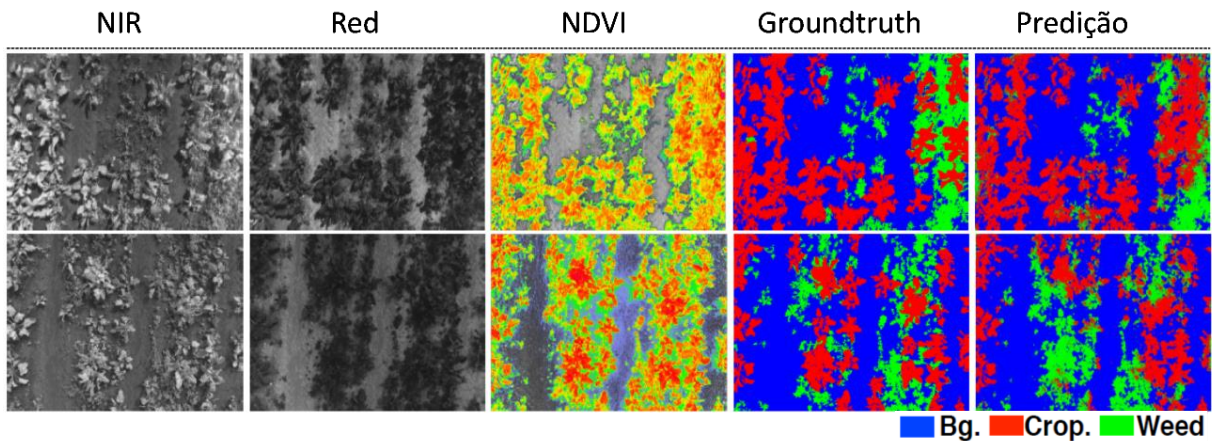
**Figura 3.13** – Desempenho de 6 modelos por classes (eixo horizontal). Valores mais altos de F1-score indicam o melhor desempenho de detecção [[SA et al. 2017](#)].

Era de se esperar que mais dados de entrada obtivessem melhores resultados, pois a rede captura mais atributos úteis que ajudam a distinguir entre as classes. Isto pode ser visto comparando-se os resultados do modelo de 2 canais (NIR + Red, barra amarela) e o modelo de 1 canal de NIR ou Red (cinza e azul-claro, respectivamente). O modelo de 2 canais supera o modelo de 1 canal para classificação de culturas e ervas daninhas. No entanto, curiosamente, o número de dados de entrada nem sempre garante a melhoria de desempenho. Por exemplo, o modelo NIR + Red supera o modelo de 3 canais (NIR + Red + NDVI) para classificação de ervas daninhas [[SA et al. 2017](#)].

### 3.3.4 Resultados qualitativos

A [Figura 3.14](#) apresenta duas instâncias dos dados de entrada multicanal. As três primeiras colunas mostram as bandas do modelo de 3 canais (NIR + Red + NDVI). O NDVI é exibido como um mapa de calor após detecção de limite (as cores mais próximas de vermelho representam uma resposta mais alta). A quarta coluna é o *groundtruth* e a quinta é a saída de probabilidade da rede. Cada probabilidade de classe com valores [0, 1 e 2] é codificada por cores, de modo que as classes de fundo, cultura e erva daninha são representadas em azul, vermelho e verde, respectivamente. Existem áreas de erro de classificação de ervas daninhas na primeira linha e erro de classificação de cultura na segunda linha. Finalmente, [Sa et al.](#) concluíram que o modelo tem menor desempenho, quando são usadas imagens de outra fase de crescimento da planta devido a variações de escala, causado por um conjunto limitado de dados de treinamento temporal de culturas e ervas daninhas de diferentes tamanhos [[SA et al. 2017](#)].





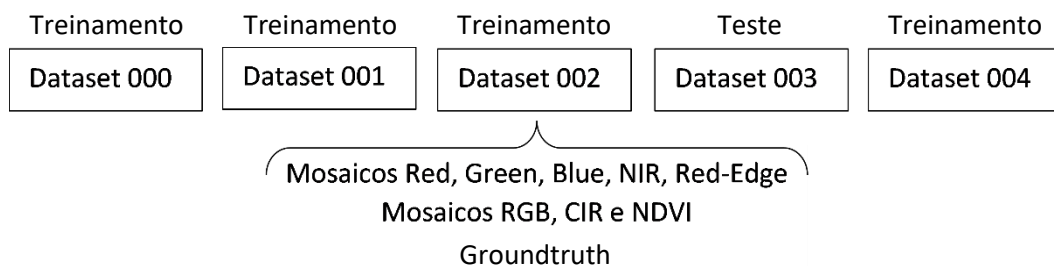
**Figura 3.14** – Resultados qualitativos de duas cenas. As três primeiras colunas são bandas de entrada da rede, com a quarta e quinta coluna mostrando groundtruth e previsões de probabilidade. A probabilidade de cada classe é codificada por cores, onde o fundo, a cultura e a erva daninha são representadas em azul, vermelho e verde, respectivamente [SA *et al.* 2017].

### 3.4 WeedMap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming

Na [Seção 3.3](#), vimos que WeedNet [SA *et al.* 2017], assim como outros estudos sobre segmentação semântica de culturas/ervas daninhas, processa apenas imagens individuais capturadas por VANT, com até três canais, devido às limitações de memória da unidade de processamento gráfico (GPU). Para produzir mapas completos de uma determinada área, SA *et al.* desenvolveram um sistema WeedMap [SA *et al.* 2018], que opera em grandes ortomosaicos com vários canais multiespectrais, usando uma rede de segmentação baseada em CNN para mapeamento semântico de plantas daninhas.

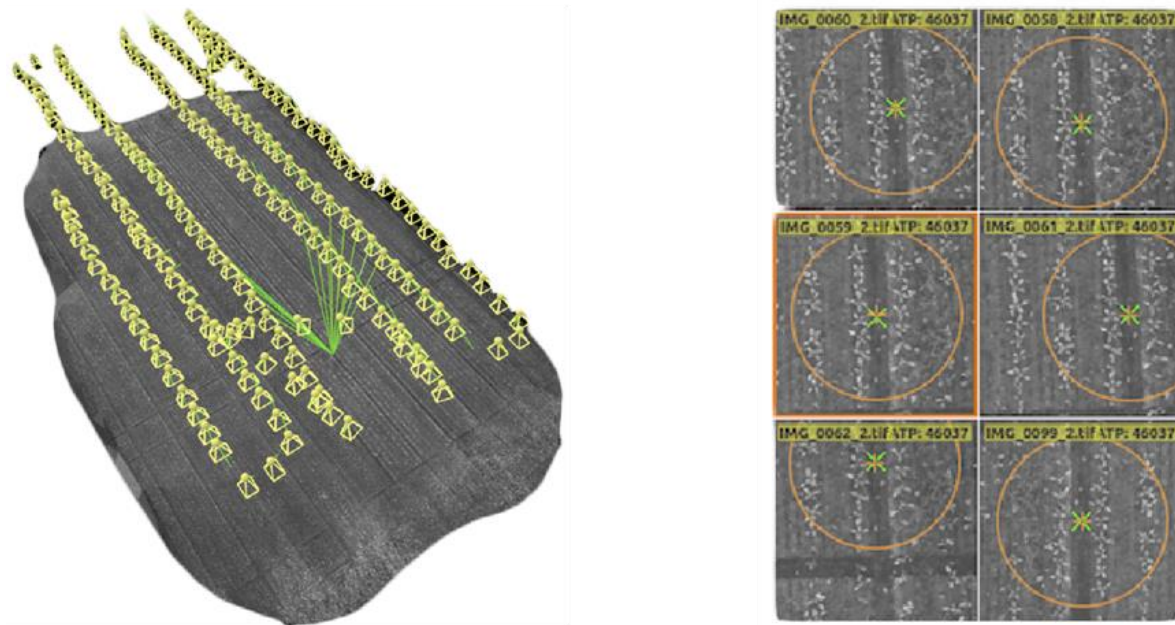
#### 3.4.1 Aquisição do conjunto de dados

Cinco conjuntos de dados multiespectrais foram coletados cobrindo diferentes plantações de beterraba com uma câmera *RedEdge-M*, que pode capturar cinco bandas espectrais (*Red*, *Green*, *Blue*, *NIR* e *Red-Edge*) perfeitamente alinhadas. Assim, para cada conjunto de dados foram gerados cinco mapas ortomosaicos multiespectrais ([Figura 3.15](#)).



**Figura 3.15** – Cinco conjunto de dados de mosaicos multiespectrais.

A [Figura 3.16](#) ilustra o ortomosaico de uma das bandas multiespectrais do conjunto de dados de treinamento #002, indicando a trajetória de voo e as poses da câmera onde as imagens multiespectrais foram registradas.



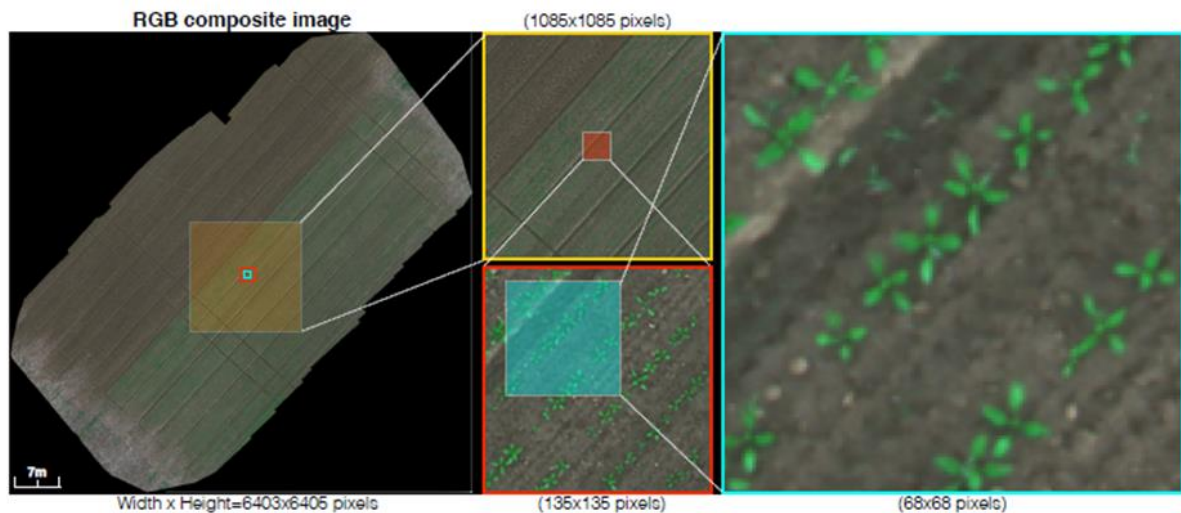
**Figura 3.16** – Mapeamento de um campo de beterraba de 1.300 m<sup>2</sup>. Cada pirâmide amarela indica a posição onde uma imagem foi tirada, e as linhas verdes são raios entre um ponto 3D e suas múltiplas visualizações [SA et al. 2018].

Cada conjunto foi anotado manualmente. Os conjuntos de dados [000, 001, 002, 004] foram usados para treinamento e o conjunto #003, para teste. Foram considerados três classes – fundo, cultura e erva daninha – identificadas numericamente por [0, 1, 2], coloridas como [*bg* = azul, *crop* = verde, *weed* = vermelho].

Os cinco ortomosaicos originais perfeitamente alinhados (*Red*, *Green*, *Blue*, NIR e *Red-Edge*) de cada conjunto de dados foram compostos para obter um ortomosaico RGB (empilhando os canais *Red*, *Green*, *Blue*) e um ortomosaico infravermelho colorido CIR (empilhando os canais *Red*, *Green* e NIR). Também foi extraído um mapa ortomosaico de índice NDVI para cada conjunto.

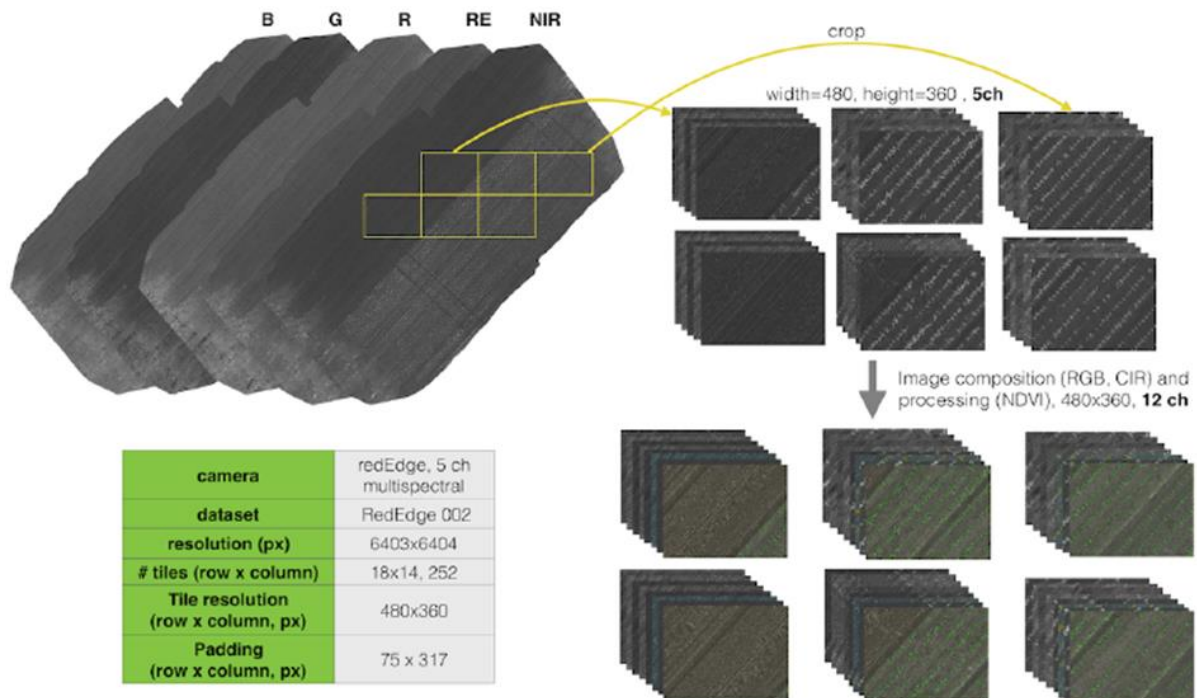
A [Figura 3.17](#) mostra um mapa ortomosaico RGB gerado a partir das bandas *Red*, *Green* e *Blue* do conjunto de dados #002 coletado pela câmera *RedEdge-M*. As caixas coloridas no ortomosaico indicam áreas de diferentes escalas no campo, que correspondem a vários níveis de *zoom*. Por exemplo, a caixa ciano na extrema direita ( $68 \times 68$  pixels) mostra uma visualização ampliada da área dentro da pequena caixa ciano no mapa ortomosaico.

Como mostrado na [Tabela 3.3](#), a composição multicanal dos ortomosaicos multiespectrais com os ortomosaicos RGB, CIR e NDVI resulta em até 12 canais, que consiste de *Red*(1), *Red-Edge*(1), *Green*(1), *Blue*(1), RGB(3), CIR(3), NDVI(1) e NIR(1), onde o número em parênteses indica o número de canais.



**Figura 3.17** – Ortomosaico RGB do conjunto de dados de treinamento #002. A imagem da esquerda mostra o mapa ortomosaico RGB, e as do meio e da direita são partes de cada área em vários níveis de zoom [SA et al. 2018].

Como os ortomosaicos multicanais são muito grandes para serem alocados em uma GPU, eles foram recortados em uma grade regular de pequenas imagens multicanais (*tiles*) do tamanho de entrada da CNN ( $480 \times 360$  pixels), sem perder sua resolução original com GSD de 1 cm/pixel (Figura 3.18). Dessa forma, evita-se a operação de redução de tamanho na entrada da rede, que degrada significativamente o desempenho da classificação ao descartar informações visuais cruciais para distinguir cultura e ervas daninhas. Esses *tiles* de até 12 canais são inseridos na entrada da rede de segmentação por meio de uma janela deslizante, até que todo o mapa ortomosaico seja coberto.



**Figura 3.18** – Tiles do ortomosaico de 12 canais, gerados a partir do conjunto de dados de treinamento #002 com cinco mosaicos multiespectrais, RGB, CIR e NDVI [SA et al. 2018].

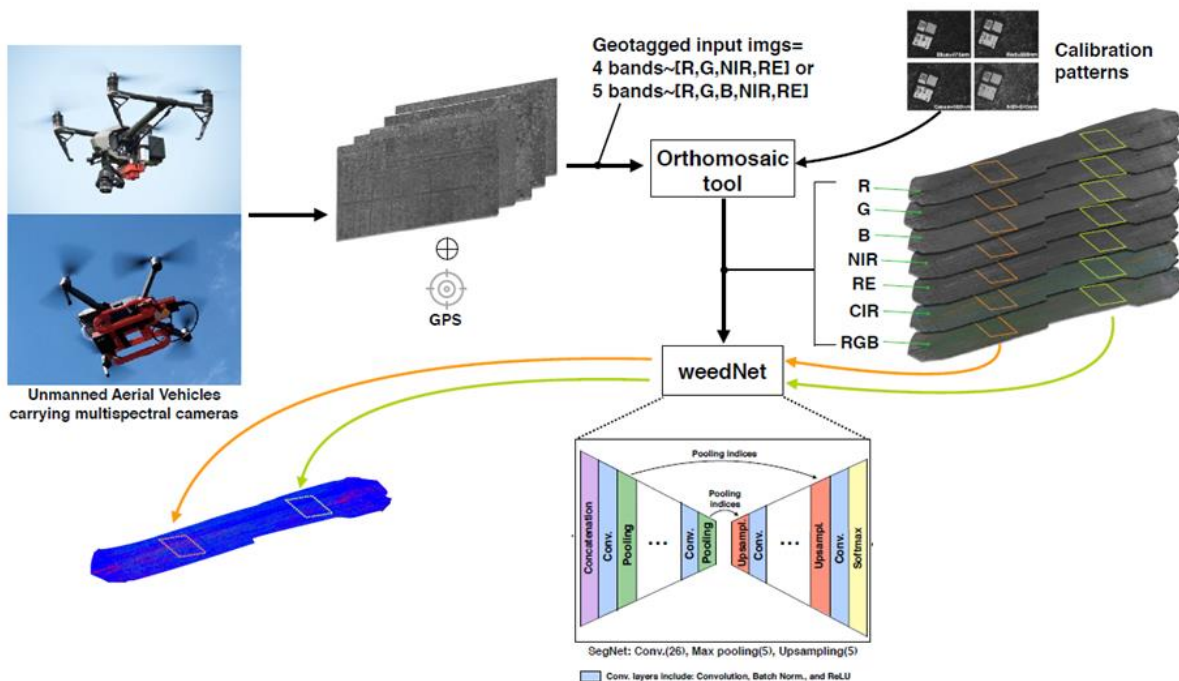


### 3.4.2 Arquitetura de segmentação semântica densa

A [Figura 3.19](#) ilustra o pipeline do WeedMap. Primeiro, imagens aéreas com cinco canais multiespectrais perfeitamente alinhadas, geolocalizadas com informações de GPS, são capturadas usando plataformas VANT, que voam a uma altura de 10 metros sobre campos de beterraba, com 80% de sobreposição lateral e frontal entre imagens consecutivas. Depois, o software aerofotogramétrico PIX4Dmapper [[PIX4DMAPPER](#)] gera mapas de reflectância (ortomosaicos multiespectrais) com calibração radiométrica, conforme discutido posteriormente no [Capítulo 4](#).

Com base nesses cinco mapas de reflectância alinhados monocanal (*Red*, *Green*, *Blue*, *NIR* e *Red-Edge*), outros mapas ortomosaicos foram compostos, como RGB(3), CIR(3) e NDVI(1). Todos os ortomosaicos originais e compostos alinhados foram recortados em uma grade regular de *tiles* de  $480 \times 360$  *pixels*, por meio de uma janela deslizante. Depois, os *tiles* multicanais foram alimentados na rede de segmentação. A saída preditiva contendo probabilidades por *pixel* para cada classe tem o mesmo tamanho da entrada, que é retornada ao local original da grade de *tiles* no mapa ortomosaico. A segmentação semântica é repetida para cada *tile* multicanal, para finalmente criar um mapa de distribuição de plantas daninhas em grande escala da área de destino.

Neste trabalho, os autores utilizaram a mesma estrutura de segmentação semântica densa utilizada no WeedNet [[SA et al. 2017](#)], ou seja, uma arquitetura SegNet-v2 [[BADRINARAYANAN et al. 2017](#)] codificador-decodificador com índices de *pooling*. Esta rede pode ser facilmente substituída por qualquer modelo de rede de segmentação densa, descrita no [Capítulo 2](#).



**Figura 3.19** – Pipeline de processamento. As imagens multiespectrais são primeiro coletadas por VANT e depois processadas para geração de ortomosaicos, que são recortados em *tiles* para entrada na rede de segmentação semântica [[SA et al. 2018](#)].

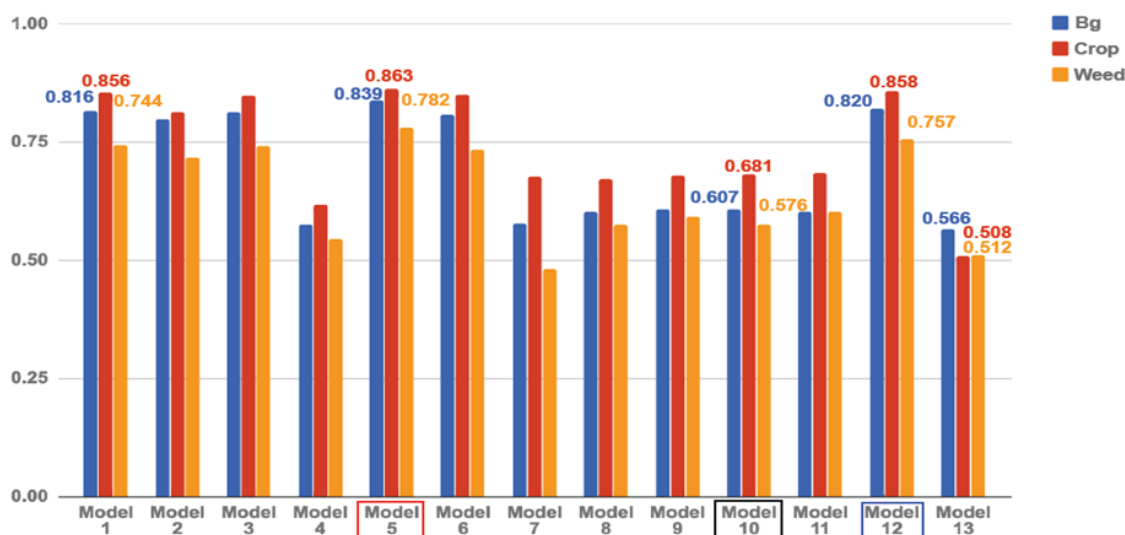
### 3.4.3 Resultados quantitativos

A [Tabela 3.3](#) mostra os resultados da segmentação densa usando 13 modelos treinados diferentes, variando o número de canais de entrada, tamanho do lote, balanceamento de classe e desempenho de cada classe (AUC – *Area Under Curve*). A fonte em negrito é usada para designar as melhores pontuações. Para melhor visualização dos resultados, a [Figura 3.20](#) mostra as pontuações AUC de cada classe.

**Tabela 3.3** – *Resumo da avaliação de desempenho com canais de entrada variados usando a câmera RedEdge-M [SA et al. 2018].*

RedEdge-M					AUC <sup>b</sup>		
Model	Canais	Multicanais <sup>a</sup>	Lotes	Bal Cls	Bg	Crop	Weed
1	12	B, CIR, G, NDVI, NIR, R, RE, RGB	6	Sim	0.816	0.856	0.744
2	12	B, CIR, G, NDVI, NIR, R, RE, RGB	4	Sim	0.798	0.814	0.717
3	12	B, CIR, G, NDVI, NIR, R, RE, RGB	6	Não	0.814	0.849	0.742
4	11	B, CIR, G, NIR, R, RE, RGB	6	Sim	0.575	0.618	0.545
5	9	B, CIR, G, NDVI, NIR, R, RE	5	Sim	<b>0.839</b>	<b>0.863</b>	<b>0.782</b>
6	9	B, G, NDVI, NIR, R, RE, RGB	5	Sim	0.808	0.851	0.734
7	8	B, G, NIR, R, RE, RGB	5	Sim	0.578	0.677	0.482
8	6	G, NIR, R, RGB	5	Sim	0.603	0.672	0.576
9	4	NIR, RGB	5	Sim	0.607	0.680	0.594
10	3	RGB (SegNet baseline)	5	Sim	0.607	0.681	0.576
11	3	B, G, R (splitted channel)	5	Sim	0.602	0.684	0.602
12	1	NDVI	5	Sim	0.820	0.858	0.757
13	1	NIR	5	Sim	0.566	0.508	0.512

(a) R, G, B, RE, NIR indicam vermelho, verde, azul, borda vermelha e canal de infravermelho próximo, respectivamente. (b) AUC é a área sob a curva.



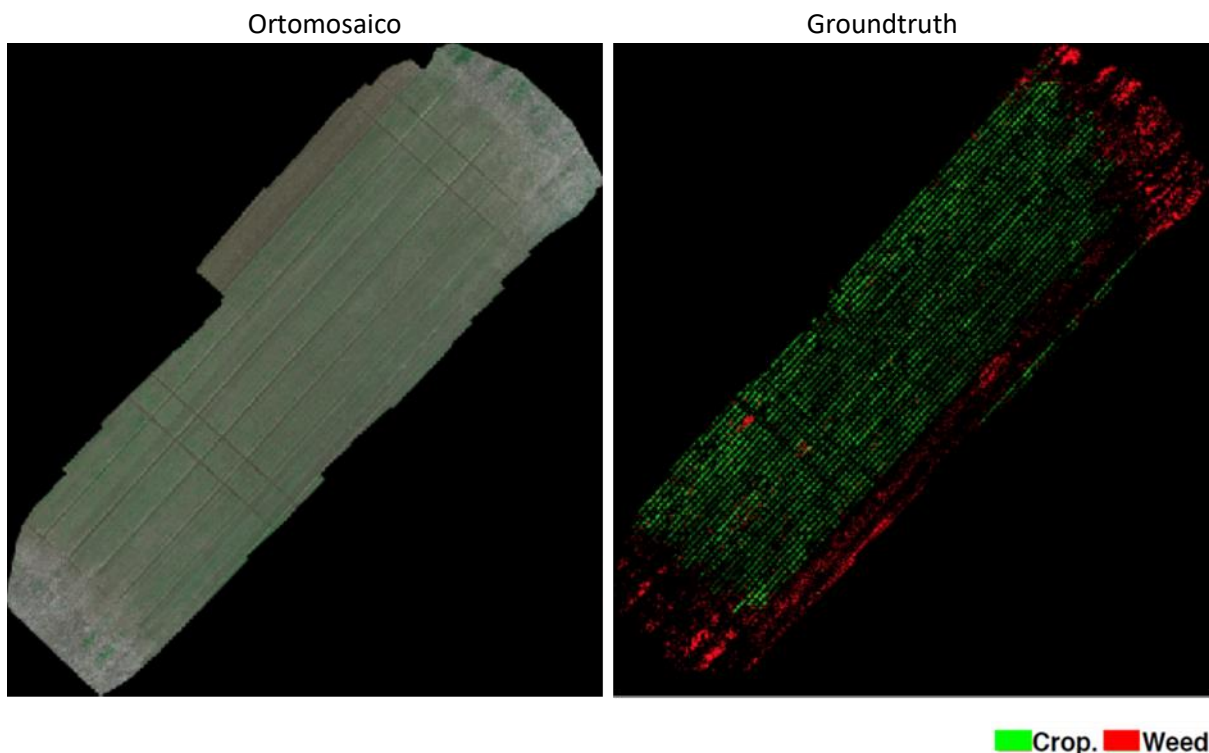
**Figura 3.20** – *Avaliação quantitativa da segmentação usando área sob a curva (AUC) do conjunto de dados RedEdge-M. O modelo 5 tem o melhor resultado, o modelo 10 é o SegNet básico com entrada de imagem RGB e o modelo 12 tem apenas uma entrada de imagem NDVI [SA et al. 2018].*

A hipótese inicial dos autores era que o desempenho melhoraria adicionando-se mais dados de treinamento, conforme evidenciado pelo modelo 1 com todas as entradas disponíveis (12 canais) e o modelo 10 (SegNet padrão com entrada RGB). No entanto, o modelo 1 com mais canais apresenta um desempenho ligeiramente inferior em comparação ao modelo 5, que tem desempenho melhor com 9 canais de entrada. O modelo 1 e o modelo 2 mostram o impacto do tamanho do lote: aumentar o tamanho do lote produz melhores resultados. O modelo 1 e o modelo 3 demonstram o impacto do balanceamento do número de imagens de cada classe. O modelo 12 supera o modelo 4, sendo que o primeiro utiliza apenas o canal NDVI, enquanto o segundo exclui apenas este canal NDVI dos 12 canais disponíveis.

De acordo com estudos anteriores, os autores concluíram que o uso do canal NDVI ajuda significativamente na discriminação de culturas e ervas daninhas, segmentando a vegetação nas imagens de entrada. Isso acontece, porque a banda NDVI já identifica a vegetação, de modo que o classificador deve apenas distinguir entre culturas e ervas daninhas. Os resultados mostraram que o NDVI contribui para uma classificação precisa da vegetação por explorar informações de entrada (NIR e *Red*) que efetivamente capturam características distinguíveis entre as classes-alvo.

### 3.4.4 Resultados qualitativos

A [Figura 3.21](#) mostra o mosaico RGB de teste #003 de alta resolução, com GSD de 1 *cm/pixel*, de um campo de beterraba sacarina com plantas daninhas, como mostrado no *groundtruth* em vermelho (daninha) e verde (cultura).



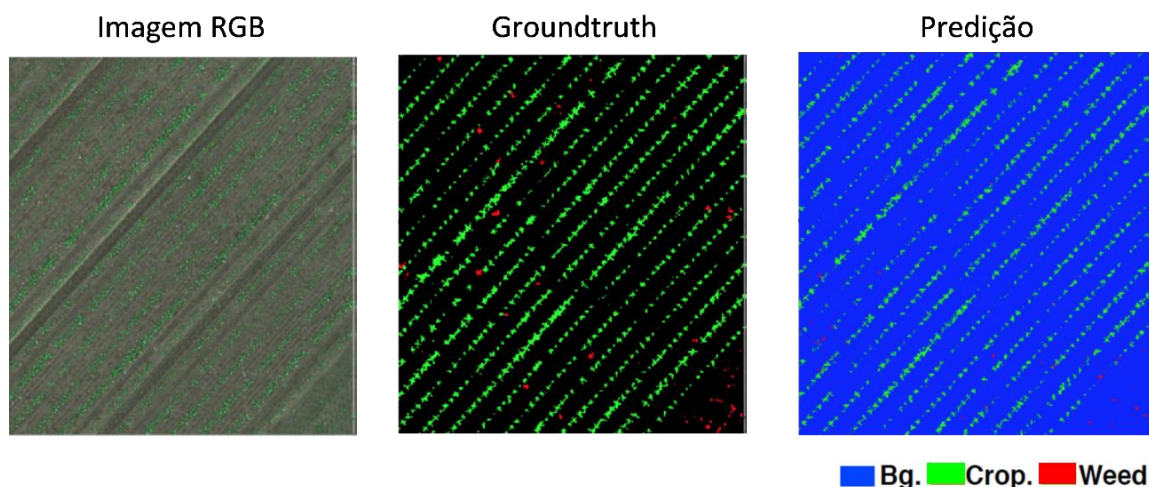
**Figura 3.21** – Conjunto de dados de teste #003 de um ortomosaico de um campo de beterraba sacarina com GSD de 1 *cm/pixel* e seu *groundtruth* correspondente [SA *et al.* 2018].



Os autores [SA *et al.* 2018] apresentaram uma análise qualitativa para o conjunto de dados de teste, mostrando que o classificador tem desempenho razoável para previsão de *pixels* da classe cultura, como mostra a Figura 3.22, com visualizações ampliadas de linhas de cultivo. No entanto, a classificação de ervas daninhas teve um desempenho relativamente inferior, em termos de falsos positivos (detecções incorretas) e falsos negativos (detecções ausentes), do que a classificação de culturas.

Os resultados qualitativos mostram, portanto, previsões boas para culturas e relativamente fracas para ervas daninhas. Isto ocorre devido a vários fatores, sendo o mais importante o fato de que a captura das imagens foi realizada em fases iniciais do crescimento das plantas, com ervas daninhas muito pequenas para serem distinguidas nas imagens, como mostrado pelo rótulo vermelho no canto inferior esquerdo da Figura 3.22. Além disso, houve um intervalo de duas semanas entre a captura do conjunto de dados de treinamento e teste, o que implica uma variação substancial no tamanho da planta.

Outro fator significativo para um menor desempenho da classe daninha é devido ao fato de que número total de *pixels* pertencentes a plantas daninhas é menor do que os da cultura, o que implica em desbalanceamento de classes, com instâncias de plantas daninhas insuficientes no conjunto de dados de treinamento. Além disso, o classificador pode estar sobreajustando ao conjunto de dados de treinamento ou aprendendo com dados de treinamento insuficientes para ervas daninhas, que representam apenas uma pequena parte de seus atributos ou características [SA *et al.* 2018].



**Figura 3.22** – Resultado qualitativo de um exemplo de imagem de entrada, *groundtruth* e previsão de saída, mostrado com um nível de zoom de 500% no mapa de ervas daninhas do ortomosaico RGB do conjunto de dados de teste 003 RedEdge-M [SA *et al.* 2018].

Portanto, a obtenção de um modelo de rede neural, que possa incorporar diferentes estágios de crescimento das plantas em vários campos agrícolas, continua sendo um desafio, especialmente devido ao fato de a aparência visual das plantas mudar significativamente durante o seu crescimento, com maior dificuldade em distinguir entre culturas e ervas daninhas no início do plantio, quando parecem semelhantes. Aumentar o conjunto de dado espaço-temporal de alta qualidade e alta resolução de VANT é necessário para resolver esse problema. No entanto, embora seja viável obter mais dados de culturas, dados representativos de ervas daninhas são mais difíceis de capturar devido à diversidade de espécies que podem ser encontradas num único campo [SA *et al.* 2018].

Nos dois trabalhos de SA *et al.*, uma questão não resolvida é o desempenho de segmentação limitado, em particular, para ervas daninhas, causado por pequenos tamanhos de instâncias de plantas e suas variações naturais em forma, tamanho e aparência. SA *et al.* [SA *et al.* 2018] argumentam que técnicas de aumento artificial de dados, como redimensionamento de imagens, rotações aleatórias e inversão horizontal, podem melhorar o desempenho da classificação. No entanto, acreditam que isso pode ser contraproducente em problemas onde as classes-alvo são visualmente semelhantes.

### 3.5 Análise e direcionamentos

Nos capítulos seguintes, apresentamos as informações sobre o trabalho desenvolvido nesta tese, levando em consideração as limitações e desafios descritos nos trabalhos relacionados desta seção, em especial, a dificuldade de detectar plantas em diferentes escalas de tamanho em imagens RGB de alta resolução.

Apesar das redes terem sido treinadas com imagens aéreas de alta resolução, testamos a rede de segmentação com um mosaico georreferenciado de imagens aéreas, em vez de imagens únicas capturadas por VANT, com a finalidade de auxiliar efetivamente no mapeamento da distribuição de plantas daninhas no campo para agricultura de precisão. Sendo assim, realizamos a rotulação de um mosaico de teste em quatro classes: solo, cultura, erva daninha e gramínea. Posteriormente, o mosaico e seu *groundtruth* são particionados em uma grade regular de *tiles* para teste da rede de segmentação.

Apesar das vantagens de utilizar mosaicos multiespectrais e mapas de índice de vegetação para gerar composições multicanais como entrada da rede de segmentação, fornecendo mais informações úteis para treinamento da rede, neste trabalho focamos, principalmente, em avaliar mais detalhadamente o desempenho das redes de segmentação semântica usando as imagens e o mosaico RGB com três canais de entrada do espectro visível devido aos desafios visuais na distinção entre cultura e ervas daninhas em imagens RGB. Nossa avaliação complementar, porém, limitada, de algumas redes usando um pequeno conjunto de imagens multiespectrais deve ser aprimorada em trabalhos futuros.

Os trabalhos abordados nas Seções 3.3 e 3.4 utilizaram a mesma rede de segmentação SegNet, variando o número de canais de entrada, com a hipótese de que quanto maior o número de canais, maior o desempenho de segmentação. Porém, apresentaram as mesmas limitações de variação em escala, principalmente, de daninhas em menor quantidade no conjunto de dados de treinamento. Nosso trabalho, por outro lado, avalia várias redes de segmentação recentes, incluindo SegNet e, no fim, implementa uma nova arquitetura que permite extrair atributos mais discriminativos de plantas daninhas em várias escalas em diferentes resoluções de entrada.

Nosso trabalho enfrenta os mesmos desafios dos trabalhos descritos neste capítulo, ou seja, poucos dados disponíveis e instâncias de plantas daninhas em menor quantidade no conjunto de dados de treinamento das redes de segmentação, que podem levar a um sobreajuste da rede. Portanto, na prática melhores resultados podem ser obtidos com um conjunto de dados mais diversificado da cultura de cana-de-açúcar, com maior variedade de ervas daninhas em diferentes estágios de crescimento.

No próximo capítulo, descrevemos sobre a plataforma VANT e processamento de imagens aéreas para obter ortomosaicos georreferenciados em aplicações de agricultura.



# Capítulo 4

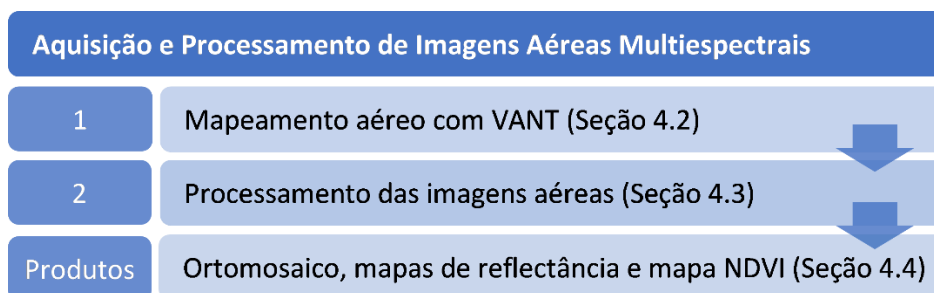
## Aquisição do conjunto de dados

Este capítulo apresenta os procedimentos para aquisição de imagens aéreas, incluindo informações sobre a plataforma VANT, especificações de uma câmera multiespectral específica para uso em aplicações de agricultura de precisão, além da descrição das etapas de pós-processamento de imagens aéreas RGB e multiespectrais para geração de ortomosaicos e mapas de reflectância georreferenciados. Por fim, apresentamos os produtos gerados após o processamento das imagens aéreas usadas neste trabalho.

### 4.1 Fluxo de trabalho

Na [Figura 4.1](#), o fluxo de trabalho para utilização de VANTs na agricultura de precisão é dividido em duas etapas: (1) mapeamento aéreo (com planejamento de voo, voo com sobreposição e captura das imagens georreferenciadas); (2) processamento de imagens aéreas para geração de um mosaico georreferenciado de alta resolução, composto de centenas de imagens cobrindo uma área maior do terreno.

A primeira etapa de mapeamento aéreo com VANT foi realizada por uma equipe do Laboratório Nacional de Ciência e Tecnologia do Bioetanol (CTBE), com a coleta de imagens aéreas de alta resolução em um campo experimental de uma cultura de cana-de-açúcar com presença de plantas daninhas. Estas imagens foram disponibilizadas para desenvolvimento deste trabalho. Na etapa seguinte, realizamos um pós-processamento das imagens para gerar um ortomosaico georreferenciado de alta resolução espacial, que possibilita a localização geográfica de plantas daninhas no campo. Para isso, usamos um software comercial de aerofotogrametria – PIX4Dmapper [[PIX4DMAPPER](#)], mas existem softwares de código aberto disponíveis – como OpenDroneMap [[OPENDRONEMAP; TOFFANIN 2023](#)] e MicMac [[RUPNIK et al. 2017](#)]. Nas seções seguintes, descrevemos os pontos relevantes destas duas etapas e mostramos os produtos gerados após o processamento das imagens aéreas coletadas no campo experimental.

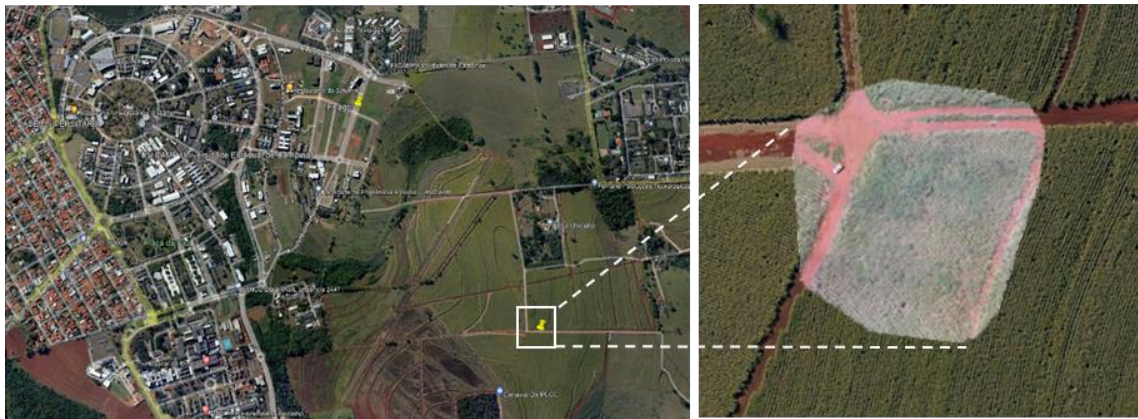


**Figura 4.1** – Etapas do fluxo de trabalho para geração de ortomosaico RGB, mapas de reflectância multiespectrais e mapa NDVI a partir de imagens aéreas de VANT.



## 4.2 Aquisição de imagens aéreas com VANT

As imagens que compõem o conjunto de dados de nosso experimento foram adquiridas no dia 06 de março de 2018, no campo experimental de uma cultura de cana-de-açúcar da Faculdade de Engenharia Agrícola (FEAGRI) da Unicamp, localizado no município de Campinas, São Paulo, Brasil, em Latitude: 22°49'26" Sul, Longitude: 47°02'60" Oeste (referência WGS84), com altitude média de 670 metros (Figura 4.2).



**Figura 4.2** – Campo experimental de cana-de-açúcar da FEAGRI-Unicamp, onde foi coletado o conjunto de imagens RGB e multiespectrais de VANT. Fonte: Google Earth.

Nesta área de aproximadamente 0,7 hectares (0,007 km<sup>2</sup>) foram registradas cana-de-açúcar, plantas daninhas de folhas largas, gramíneas de folhas estreitas e estradas de terra. A Figura 4.3 mostra detalhes de uma das imagens aéreas do campo experimental de cana-de-açúcar, contendo mamona (*Ricinus comunis*) – um arbusto verde-azulado de até 3 m de altura – com folhas grandes de 10 a 40 cm de largura, podendo chegar a 60 cm, divididas em sete a dez lóbulos.

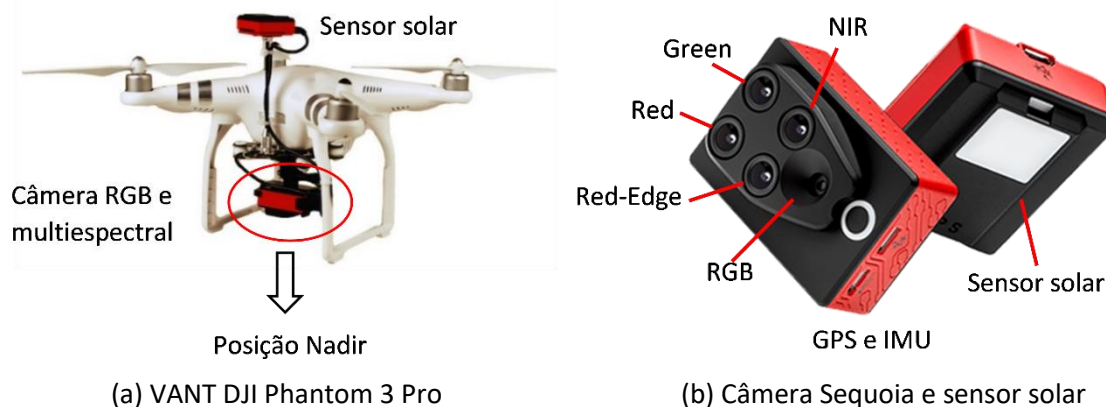


**Figura 4.3** – Plantas daninhas de folhas largas encontradas no campo experimental de cana-de-açúcar. (a) Recorte de 600 × 450 pixels de uma das imagens aéreas de VANT, mostrando uma folha de mamona de aproximadamente 40 cm de largura, dentro do retângulo amarelo de 40 × 40 pixels. (b) Mamona (*Ricinus comunis*).

Podemos perceber a dificuldade de discriminação entre plantas cultivadas e ervas daninhas devido às suas similaridades visuais e espectrais, uma vez que elas refletem comprimentos de onda eletromagnéticos similares quando captados pelos sensores RGB. Este desafio pode ser superado por VANTs com sensores multiespectrais projetados especialmente para uso na agricultura, que captam imagens em faixas espectrais visíveis e infravermelhas. Com isso, é possível obter maior precisão na detecção de ervas daninhas, por meio de imagens multiespectrais de alta resolução capturadas próximas ao solo.

#### 4.2.1 Especificação da plataforma, câmera e sensores

O campo experimental foi imageado usando uma plataforma VANT multirrotor quadricóptero de baixo custo, modelo DJI Phantom 3 Professional (Figura 4.4a), equipado com uma câmera multiespectral Micasense Parrot Sequoia com sensor solar (Figura 4.4b). A Tabela 4.1 mostra a especificação dos sensores desta câmera<sup>4</sup>.



**Figura 4.4** – VANT multirrotor quadricóptero DJI Phantom 3 Professional equipado com câmera multiespectral na posição nadir (a); Sensor solar e câmera multiespectral Sequoia com cinco sensores óticos embutidos (RGB e quatro bandas multiespectrais – Red, Green, Red-Edge e Near Infrared) (b).

**Tabela 4.1** – Especificação dos sensores da câmera Sequoia.

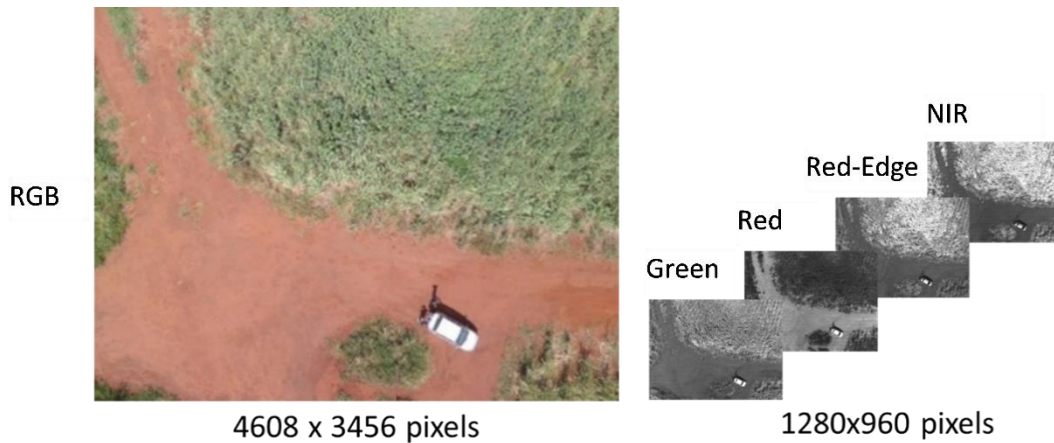
Descrição	Sensor RGB	Multiespectral	Unidade
Resolução da câmera	16	1,2	MP
Resolução da imagem	4.608 × 3.456	1.280 × 960	pixel
Tamanho do pixel no sensor	1,34	3,75	μm
Distância focal	4,88	3,98	mm
Profundidade de bits ( <i>bit depth</i> ) <sup>5</sup>	8	10	bits
Tamanho do sensor	6,17 × 4,63	4,8 × 3,6	mm
HFOV – VFOV – DFOV	63,9 – 50,1 – 73,5	61,9 – 48,5 – 73,7	graus
Obturador	Rolling shutter	Global shutter	
#bandas espectrais	3 (RGB)	4 (monocromáticas)	
Green	-	550 (40 de largura)	nm
Red	-	660 (40 de largura)	nm
Red-Edge	-	735 (10 de largura)	nm
Near Infrared (NIR)	-	790 (40 de largura)	nm

<sup>4</sup> Sensores adicionais embutidos na câmera: GPS (*Global Positioning System*) e IMU (*Inertial Measurement Unit*) para registrar coordenadas de posição e atitude (ângulos de rotação nos três eixos) do VANT, respectivamente.



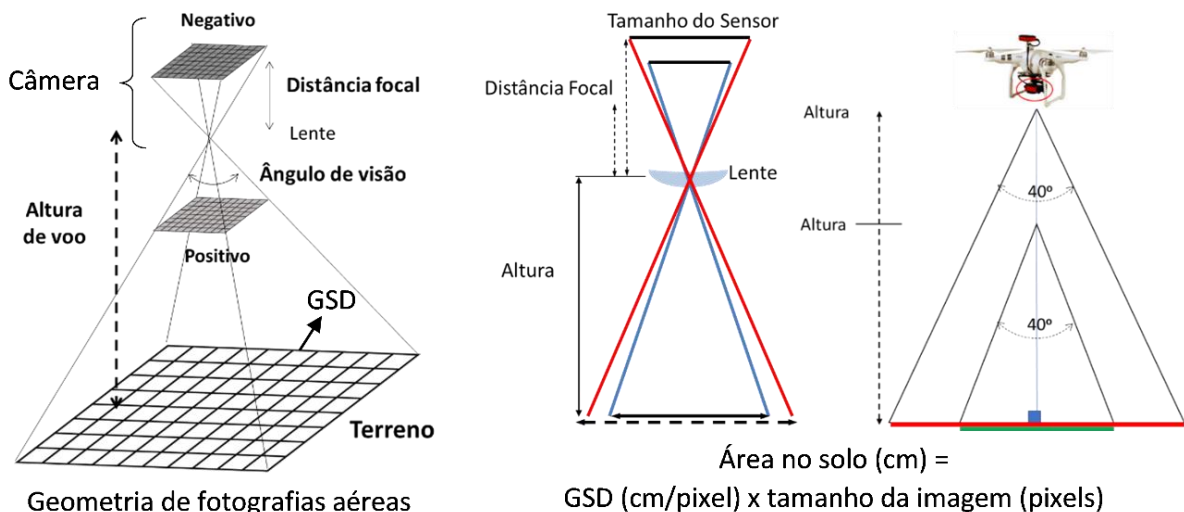
## 4.2.2 Resolução espacial das imagens aéreas

A câmera Sequoia captura imagens RGB no espectro visível (R, G e B), com uma resolução de  $4.608 \times 3.456$  *pixels*, usando um sensor de 16 *megapixels* e radiometria<sup>5</sup> de 8 bits de resolução para cada canal RGB. Além disso, quatro imagens multispectrais de um canal – nas bandas espectrais vermelho (*Red*), verde (*Green*), borda do vermelho (*Red-Edge*) e infravermelho próximo (NIR) – são capturadas com uma resolução de  $1.280 \times 960$  *pixels*, usando quatro sensores espectrais de 1,2 *megapixels* e radiometria de 10 bits de resolução (Figura 4.5).



**Figura 4.5** – Resolução das imagens dos sensores da câmera multispectral Sequoia.

Em sensoriamento remoto, a resolução espacial ou GSD (*Ground Sample Distance*) de uma imagem aérea e a cobertura de área no terreno são afetados pela resolução da imagem em *pixels*, altura de voo, tamanho do sensor e distância focal da lente, como ilustrado na Figura 4.6. O GSD é a distância entre dois *pixels* consecutivos medidos no solo. Quanto maior o valor do GSD, menor a resolução espacial e, portanto, menos detalhes são visíveis na imagem [PIX4DMANUAL].



**Figura 4.6** – Fatores que influenciam no GSD e área de cobertura da imagem no solo. Baseado em [BRITO e COELHO 2012; WOLF *et al.* 2014; JENSEN 2009].

<sup>5</sup> Resolução radiométrica: sensibilidade de um sensor para distinguir dois níveis de intensidade de refletância recebida. Se refere ao número de divisões de profundidade de bits (por exemplo, 255 níveis digitais para 8 bits, 1.024 para 10 bits, 65.536 para 16 bits, etc.) nos dados coletados por um sensor.

De acordo com a [Equação 4.1](#), o GSD é proporcional à altura de voo, onde 100 é um fator de escala para obter o GSD em  $\text{cm/pixel}$  [[PIX4DMANUAL](#)]. Assim, em alturas de voo de 30 a 150 m, o sensor RGB da câmera Sequoia captura imagens com GSD de 0,8 cm a 4,1 cm, enquanto as imagens multiespectrais apresentam um GSD de 2,8 cm a 14,1 cm. Desta forma, é possível determinar a altura de voo necessária para obter um GSD adequado aos objetivos específicos de cada aplicação.

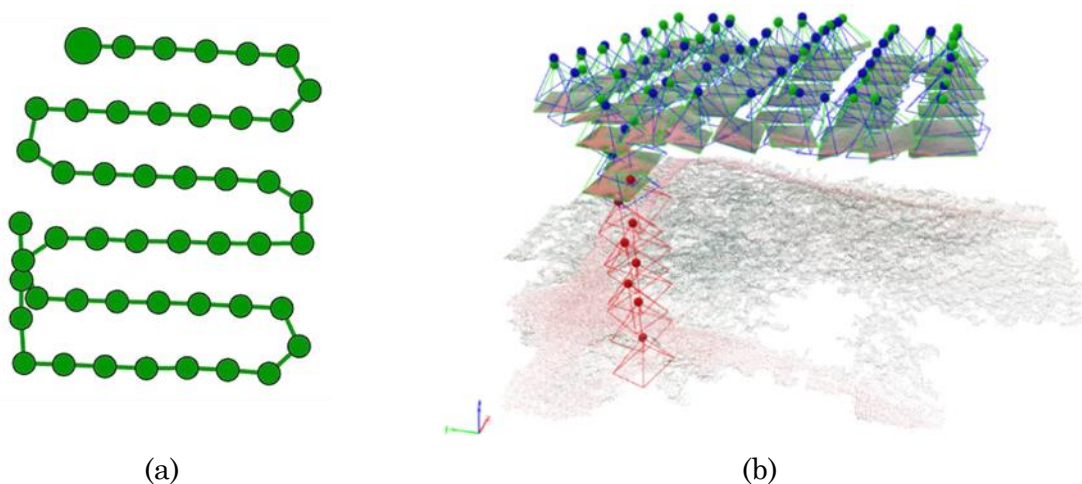
$$\text{GSD} = \frac{\text{Altura (metros)} \times \text{Largura do sensor (milímetros)} \times 100}{\text{Distância focal (milímetros)} \times \text{Largura da imagem (pixels)}} \quad (4.1)$$

Em aplicações de mapeamento da distribuição de plantas daninhas no campo, uma característica indispensável das imagens é a alta resolução espacial, que pode ser alcançada com voos mais baixos. Neste caso, para determinar a altura ideal, o GSD deve ser especificado a partir do tamanho das mudas de plantas daninhas a serem discriminadas, a distância entre as linhas da cultura e o tipo de cultura.

Assim, as imagens foram capturadas com uma altura de 40 metros em relação ao solo, proporcionando imagens RGB e multiespectrais com GSD médio de aproximadamente 1,16 e 4,63  $\text{cm/pixel}$ , respectivamente. A altura de voo e, conseqüentemente, o GSD de cada imagem não tiveram variação significativa ao longo do mapeamento devido ao terreno plano do campo experimental de cana-de-açúcar.

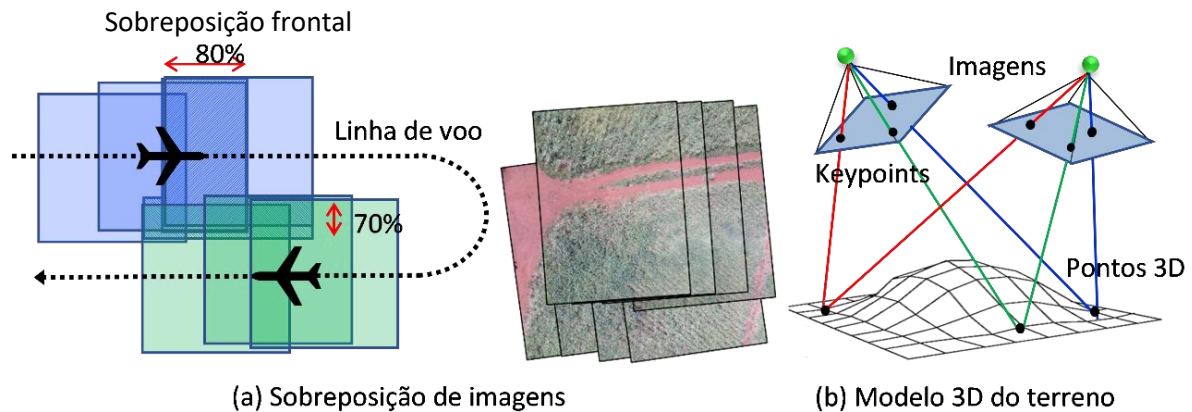
### 4.2.3 Conjunto de imagens aéreas

O disparo da câmera é ativado automaticamente pelo VANT de acordo com um plano de voo programado. A [Figura 4.7a](#) mostra a trajetória de voo do VANT durante o mapeamento aéreo do campo experimental de cana-de-açúcar, indicando as coordenadas geográficas onde as imagens foram registradas.



**Figura 4.7** – Plano de voo do campo experimental de cana-de-açúcar realizado por um VANT, onde cada ponto verde indica a posição geográfica ( $X$ ,  $Y$ ,  $Z$ ) da imagem capturada (a); cada pirâmide azul indica a posição e orientação da câmera onde uma imagem foi capturada, sendo descartadas as pirâmides vermelhas correspondentes à decolagem (b).

Para imagens agrícolas, é recomendável um plano de voo com alta sobreposição frontal (em relação à direção de voo) e lateral (entre linhas paralelas de voo) entre duas imagens, como mostra a [Figura 4.8a](#) [PIX4DMANUAL]. Isso permite obter pontos-chave homólogos suficientes entre duas ou mais imagens para reconstrução do modelo 3D do terreno [WESTOBY *et al.* 2012], usando os conceitos de estereoscopia ilustrado na [Figura 4.8b](#). Quanto maior a área de sobreposição, melhor a precisão do modelo. A partir da sobreposição, altura de voo e da especificação do sensor da câmera, um software de planejamento de voo calcula o número de imagens necessário para cobrir toda a área.



**Figura 4.8** – Sobreposição frontal e lateral para obtenção de keypoints 2D em cada imagem aérea (a), para reconstrução de pontos 3D no terreno por estereoscopia (b). Baseado em [BRITO e COELHO 2012; JENSEN 2009].

No mesmo voo, definido para capturar imagens com sobreposição frontal de 80% e lateral de 70%, a 40 metros de altura do solo, foram coletados: a) um conjunto de 63 imagens RGB, em formato JPEG com compressão; e b) um conjunto de 63 imagens de cada banda multiespectral, em formato GeoTIFF sem compressão, como listado na [Tabela 4.2](#).

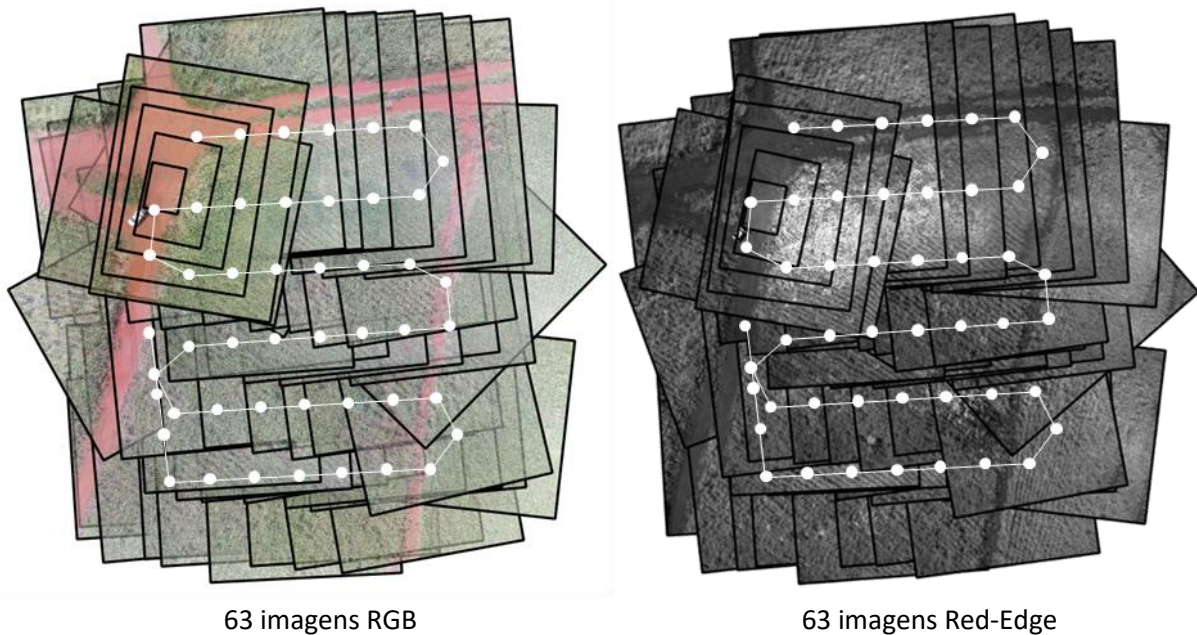
Os valores de cada *pixel* da imagem RGB são representados com resolução de 8 bits (256 níveis) para cada canal de cor. No entanto, os sensores multiespectrais têm maior sensibilidade e capturam imagens com valores de radiância com resolução radiométrica de 10 bits (1.024 níveis) armazenados em 16 bits. No pós-processamento das imagens aéreas multiespectrais para geração do mosaico multiespectral georreferenciado em formato GeoTIFF, estes valores serão calibrados radiometricamente com uso do sensor solar, gerando valores de reflectância entre 0 e 1, armazenados em ponto flutuante de 32 bits.

**Tabela 4.2** – Conjuntos de imagens aéreas do campo experimental de cana-de-açúcar.

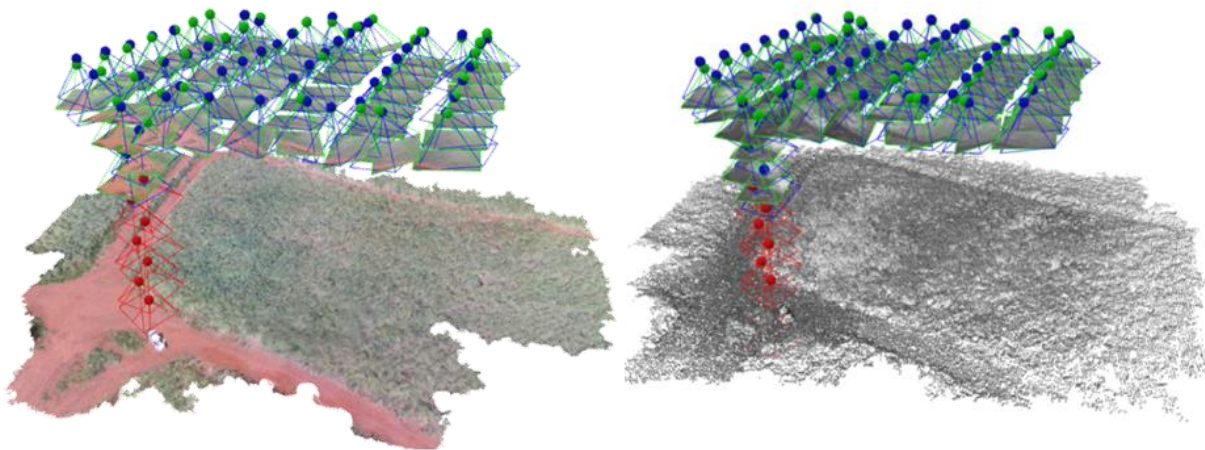
Imagens (JPEG) GeoTIFF	Número imagens (calibradas)	Largura x altura (pixels)	Bandas	Tipo de dados	Sobreposição frontal   lateral (%)	Altura de voo (m)	GSD médio (cm/pixel)
(RGB)	63 (48)	4.608 × 3.456	3	Byte	80   70	40	1,16
Green	63 (48)	1.280 × 960	1	UInt16	80   70	40	4,63
Red	63 (48)	1.280 × 960	1	UInt16	80   70	40	4,63
Red-Edge	63 (48)	1.280 × 960	1	UInt16	80   70	40	4,63
NIR	63 (48)	1.280 × 960	1	UInt16	80   70	40	4,63



A [Figura 4.9a](#) mostra as imagens aéreas geolocalizadas com coordenadas do GPS, sobrepostas ao longo da trajetória do voo, sem nenhum processamento para correção de distorções de perspectiva. As imagens capturadas no início do voo ([Figura 4.9b](#)) têm altura variável e, conseqüentemente, GSDs diferentes. Portanto, as imagens iniciais do voo devem ser descartadas de cada conjunto, sendo que 48 das 63 imagens capturadas foram efetivamente calibradas no pós-processamento (ou seja, tiveram sua posição e orientação inicial ajustadas), podendo ser usadas para geração do mosaico georreferenciado.



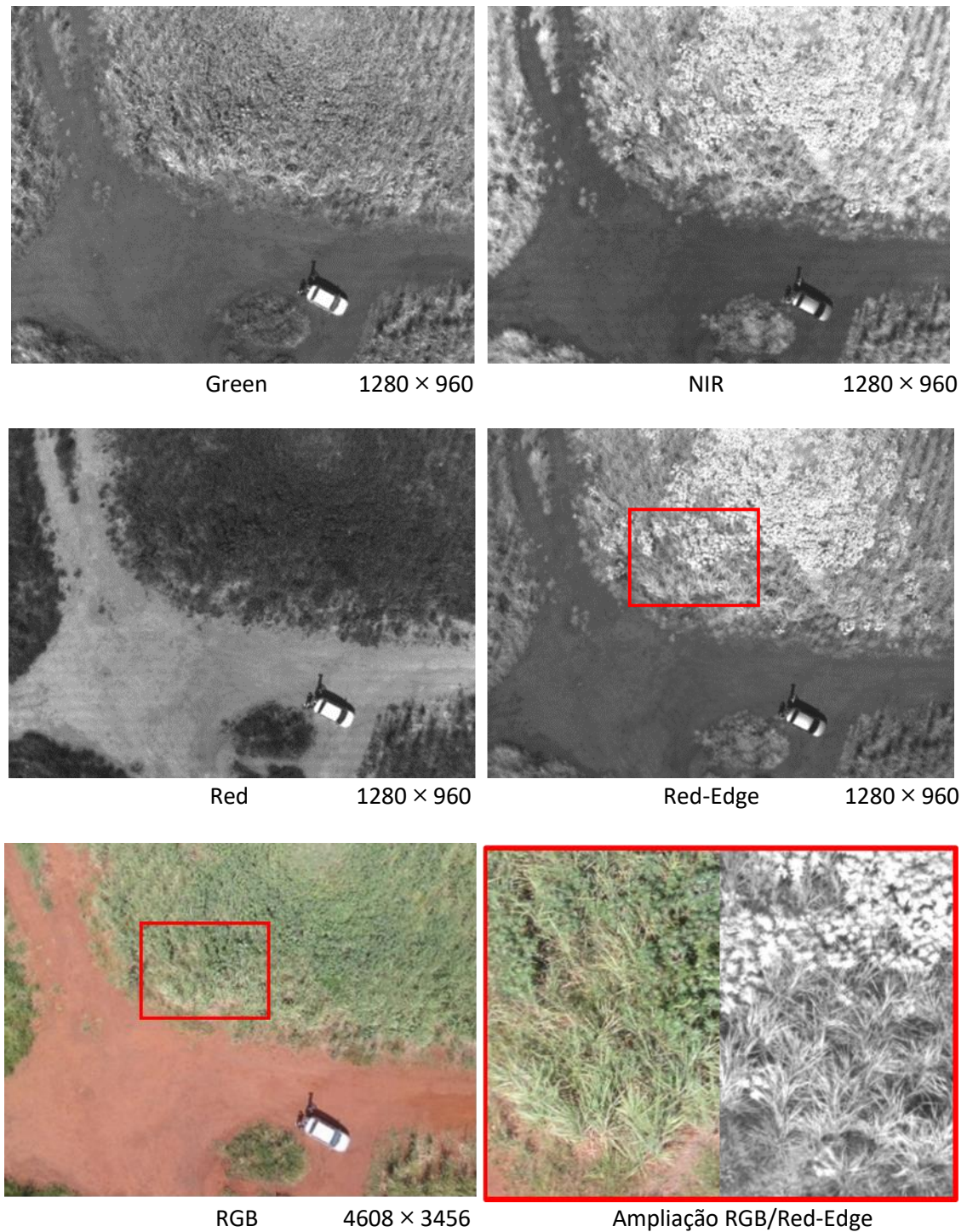
(a) Justaposição das imagens sobrepostas sem correções de perspectiva.



(b) Posição geográfica das imagens capturadas sobre o terreno 3D.

**Figura 4.9** – Conjuntos de imagens aéreas RGB e Red-Edge capturadas com sobreposição frontal e lateral na mesma trajetória de voo, mostradas sem correção de distorções de perspectiva (a). Visualização das posições geográficas das imagens capturadas (pirâmides azuis) e imagens iniciais do voo descartadas (pirâmides vermelhas) sobre o terreno 3D (b).

A [Figura 4.10](#) mostra imagens de quatro bandas multiespectrais e a imagem RGB capturadas no mesmo instante pela câmera Sequoia. A imagem inferior direita mostra os detalhes ampliados das imagens dos sensores RGB e *Red-Edge*, com destaque para plantas daninhas de folhas largas em cultura de cana-de-açúcar. De forma geral, a câmera multiespectral apresenta uma maior capacidade de discriminação entre cultura e plantas daninhas do que as imagens RGB.



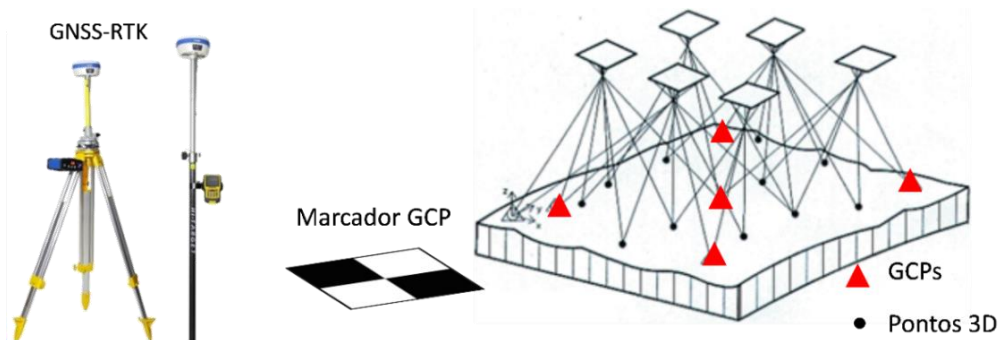
**Figura 4.10** – Uma imagem de cada banda multiespectral e do espectro visível capturada pela câmera Sequoia. A última imagem apresenta uma visão ampliada da área dentro da caixa vermelha, mostrando detalhes que sugerem os desafios visuais na distinção entre cultura e ervas daninhas em imagens RGB. *Fonte: Imagens capturadas pelo CTBE.*



#### 4.2.4 Conjunto de pontos de controle georreferenciados

A posição inicial de cada imagem no ponto de disparo é definida pelas coordenadas geográficas dadas por um sistema GPS (*Global Positioning System*) de baixa precisão integrado na câmera. Para melhorar a precisão de georreferenciamento do mosaico, geralmente são coletados dados adicionais, como por exemplo, pontos de controle no terreno (*Ground Control Point – GCP*) com coordenadas geográficas precisas da ordem de poucos centímetros (Figura 4.11). Para isso, pontos de referência identificáveis nas fotografias (pontos naturais ou marcadores implantados no solo) são medidos por um equipamento GNSS-RTK (*Global Navigation Satellite System – Real Time Kinematic*) de alta precisão para auxiliar na geração de mapas ortomosaicos georreferenciados com precisão centimétrica [WOLF *et al.* 2014; JENSEN 2009]. Uma alternativa seria usar uma tecnologia moderna de VANTs com GNSS-RTK embutido.

Assim, para aplicação deste trabalho em agricultura de precisão, seria necessário coletar GCPs ou usar um VANT com RTK para gerar os mosaicos georreferenciados com maior precisão. Porém, não foram coletados GCPs no levantamento de campo da área experimental de cana-de-açúcar. No entanto, apesar do ortomosaico georreferenciado gerado não permitir a localização geográfica de objetos com precisão centimétrica no terreno, as posições e orientações iniciais das imagens aéreas são corretamente ajustadas por softwares aerofotogramétricos para reconstrução do modelo 3D do terreno e geração do mosaico, mesmo sem auxílio de GCPs. A coleta de GCPs e georreferenciamento preciso do mosaico podem ser realizados posteriormente. Desta forma, esta limitação não prejudica o trabalho de segmentação semântica de plantas daninhas.



**Figura 4.11** – GNSS-RTK para obtenção das coordenadas geográficas de alta precisão dos pontos de controle (GCPs) em marcadores distribuídos na área de interesse do terreno.

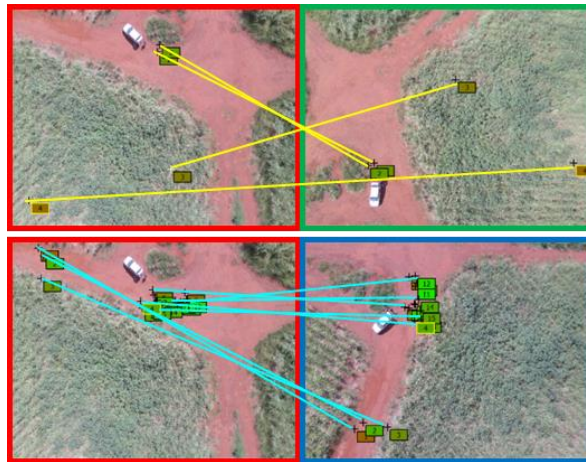
### 4.3 Etapas de processamento de imagens aéreas

A maioria das aplicações práticas em agricultura requer mapas que cobrem grandes áreas (na ordem de dezenas de hectares), preservando os detalhes da distribuição das plantas no campo. Embora os sensores a bordo de VANTs forneçam imagens de alta resolução espacial, cada imagem cobre apenas uma pequena área de interesse. Assim, um pós-processamento de imagens é necessário para gerar um ortomosaico de alta resolução, com maior cobertura de área [BRITO e COELHO 2012; WOLF *et al.* 2014; JENSEN 2009].

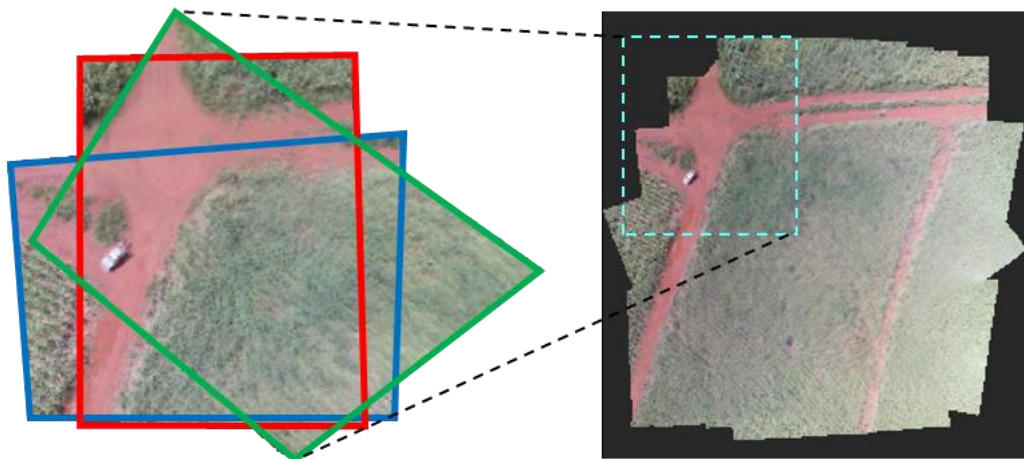
Alguns softwares de geração de mosaico (ou panorâmica) utilizam o algoritmo SIFT (*Scale Invariant Feature Transform*) [LOWE 2004], que detecta pontos-chave (*keypoints*) nas imagens, e RANSAC [FISCHLER e BOLLES 1981], que faz a correspondência dos pontos homólogos de duas ou mais imagens, para uní-las em um mosaico (Figura 4.12).



Durante o processo de união, as imagens passam por transformações geométricas afins (translação, escala, rotação, *shear*), para se encaixarem adequadamente (Figura 4.13). No entanto, esses ajustes ocasionam uma distorção acumulada ao longo do processo, que deve ser corrigida [BROWN e LOWE 2007].



**Figura 4.12** – Detecção de keypoints e correspondência de características.



**Figura 4.13** – Geração de mosaico por detecção/correspondência de keypoints e transformações geométricas das imagens com o software Kolor Autopano Giga 4.4.

Apesar dos resultados visuais destes softwares mais simples serem satisfatórios, na agricultura de precisão é necessário obter mosaicos com precisão geométrica, e, para isso, softwares específicos de aerofotogrametria são empregados para geração do mosaico de imagens aéreas de VANT, utilizando uma técnica de visão computacional, chamada *Structure from Motion* (SfM) [HARTLEY, ZISSERMAN 2003], aliada aos conceitos de aerofotogrametria [WOLF *et al.* 2014; JENSEN 2009].

Além de softwares comerciais de aerofotogrametria, tais como PIX4Dmapper [PIX4DMAPPER] e Agisoft MetaShape (PhotoScan) [agisoft.com], há opções de código aberto, como OpenDroneMap [OPENDRONEMAP; TOFFANIN 2023] e MicMac [RUPNIK *et al.* 2017]. Após o mapeamento aéreo realizado pelo CTBE, usamos o PIX4Dmapper para geração de ortomosaicos georreferenciados, onde cada *pixel* do mosaico tem suas coordenadas conhecidas num dado sistema de referência geográfica e corresponde a uma unidade métrica.

O processamento de imagens aéreas usando PIX4Dmapper consiste em quatro etapas [PIX4DMANUAL]: (1) processamento inicial com aerotriangulação para orientação das imagens e geração da nuvem de pontos 3D esparsa; (2) densificação da nuvem de pontos 3D e geração de uma malha de superfície 3D a partir da nuvem de pontos densificada; (3) geração do modelo digital de superfície (MDS); e (4) ortorretificação do mosaico ou mapas de reflectância, pela projeção das imagens RGB e multiespectrais no MDS. Nesta última etapa, também podem ser gerados mapas de índices de vegetação. A seguir, descrevemos as etapas de processamento de imagens e as saídas geradas, como resumido na Figura 4.14.

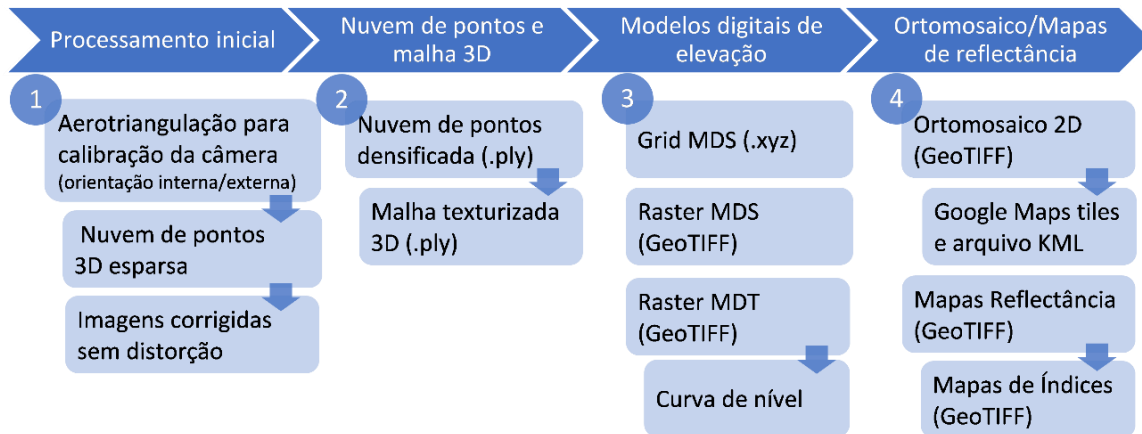


Figura 4.14 – Etapas de processamento de imagens e saídas geradas no PIX4Dmapper.

### 4.3.1 Etapa 1 – Processamento inicial

Para iniciar um projeto no PIX4Dmapper, importamos os arquivos das imagens RGB em formato JPEG com compressão ou das imagens multiespectrais em formato TIFF sem compressão (Tabela 4.2). Os metadados no EXIF das imagens são carregados automaticamente, com informações de orientação interior da câmera (centro de projeção CP, distância focal  $f$ , tamanho do sensor, etc.) e orientação exterior inicial (posição geográfica do GPS embutido de baixa precisão e ângulos de orientação da câmera dado pelo sensor IMU) [BRITO e COELHO 2012; JENSEN 2009] (Figura 4.15).

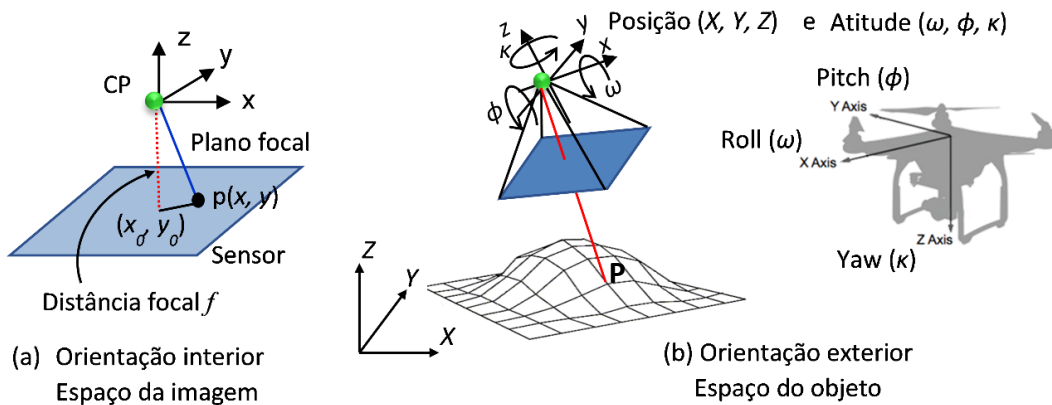
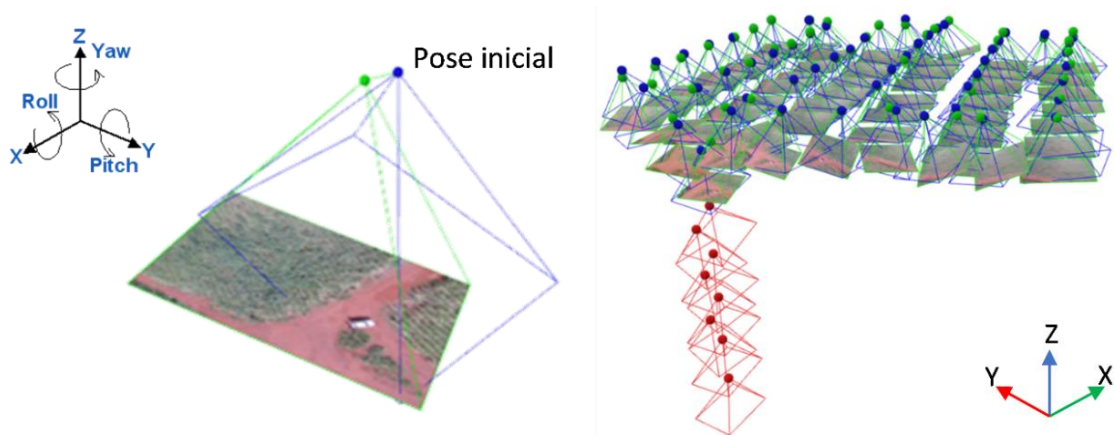


Figura 4.15 – (a) Orientação interior da câmera (distância focal, etc.); (b) Orientação exterior das imagens: posição  $(X, Y, Z)$  e atitude do VANT (roll, pitch, yaw –  $\omega, \phi, \kappa$ ).

No início desta etapa de processamento, as imagens aéreas são usadas para realizar as seguintes tarefas [PIX4DMANUAL]:

- **Orientação da câmera:** determina a orientação interna (distância focal, centro de projeção, etc.) e orientação externa da câmera em relação ao solo (posição e orientação inicial) dados pelo EXIF da imagem [WOLF *et al.* 2014];
- **Extração de *keypoints*:** detecta milhares de *features* nas imagens (pontos-chaves) pelo método SIFT [LOWE 2004] e encontra *keypoints* correspondentes (pontos homólogos ou *matched points*) nas áreas de sobreposição entre duas ou mais imagens, usando o método RANSAC [FISCHLER e BOLLES 1981].

Inicialmente, as posições geográficas (X, Y, Z) e a atitude do VANT (ângulos de rotação dos três eixos – *roll*, *pitch* e *yaw*) no momento da captura (pirâmide azul) são usados para automaticamente orientar as imagens no espaço-objeto. Nesta primeira etapa, as poses das imagens (posição e orientação) são ajustadas pelo software aerofotogramétrico (pirâmide verde) [WOLF *et al.* 2014]. Se houver erro na calibração de uma imagem, ela é desabilitada (*frustum* vermelho), não sendo usada no restante do processamento (Figura 4.16).



**Figura 4.16** – Poses iniciais das imagens aéreas (azul) e poses ajustadas (verde).

Após o posicionamento inicial das imagens no espaço com informações do GPS embutido, a extração automática de pontos-chave (*keypoints*) e a correspondência de pontos homólogos nas múltiplas imagens sobrepostas são executadas (Figura 4.17). Quanto maior a área de sobreposição, mais *keypoints* podem ser encontrados, e, portanto, mais precisamente os pontos 3D no modelo do terreno serão calculados (Figura 4.18).

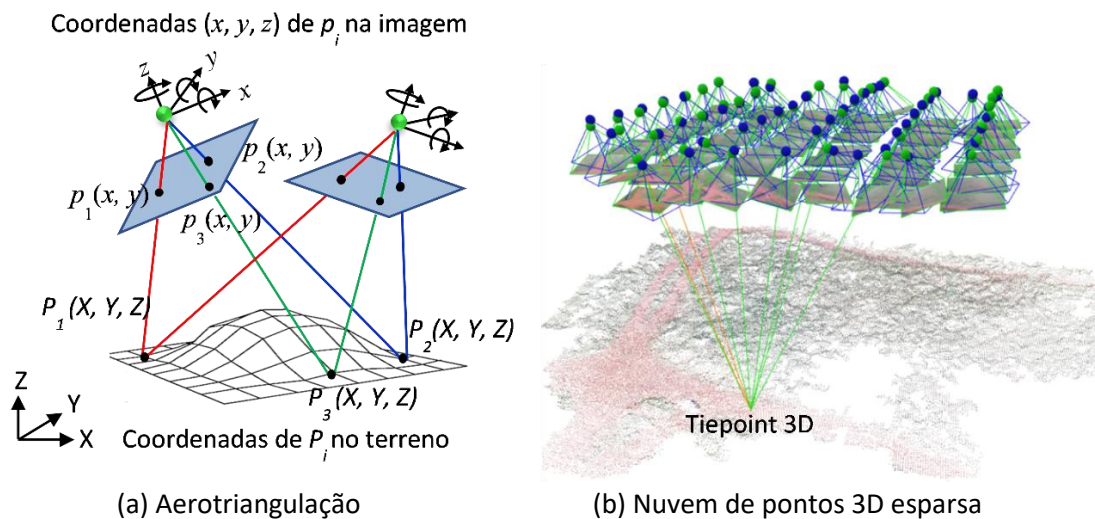


**Figura 4.17** – Sobreposição de imagens para extração de keypoints correspondentes em múltiplas imagens e pontos-chave encontrados em uma imagem RGB.



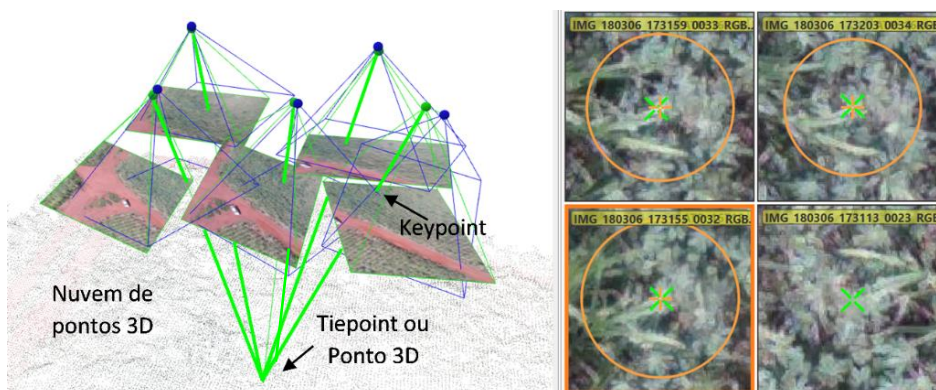
Tendo disponíveis as informações dos parâmetros intrínsecos da câmera (distância focal, etc.) e das coordenadas dos *keypoints* homólogos na imagem, os parâmetros extrínsecos (pose) da câmera podem ser calculados. Assim, todas as imagens são calibradas, ou seja, orientadas no espaço, com rotações e posições iniciais da câmera ajustadas em relação ao sistema de coordenadas do objeto [WOLF *et al.* 2014]. Em seguida, as coordenadas dos pontos 3D (*tiepoints*) no espaço-objeto são calculadas por aerotriangulação (Figura 4.18a) para gerar uma nuvem de pontos esparsa (Figura 4.18b).

Softwares modernos de aerofotogrametria usam o método *Structure from Motion* [WESTOBY *et al.* 2012] para reconstrução automática dos *tiepoints* por estereoscopia, com base nos *keypoints* homólogos. Ao contrário de métodos tradicionais de aerofotogrametria, onde primeiro a orientação externa das imagens deve ser determinada usando GCPs com coordenadas conhecidas no espaço-imagem e espaço-objeto, para posterior reconstrução dos pontos 3D no terreno, o SfM realiza a aerotriangulação com um ajuste global (*bundle adjustment* – BA) [TRIGGS *et al.* 2000], que otimiza simultaneamente os parâmetros externos da câmera e as coordenadas dos *tiepoints* [OPENDRONEMAP], sem nenhum GCP.



**Figura 4.18** – Geração de nuvem de pontos 3D esparsa usando aerotriangulação (SfM). *Tiepoints* ( $P_i$ ) são pontos 3D no terreno visualizados em várias imagens, que correspondem aos *keypoints* homólogos ( $p_i$ ) detectados automaticamente no espaço da imagem.

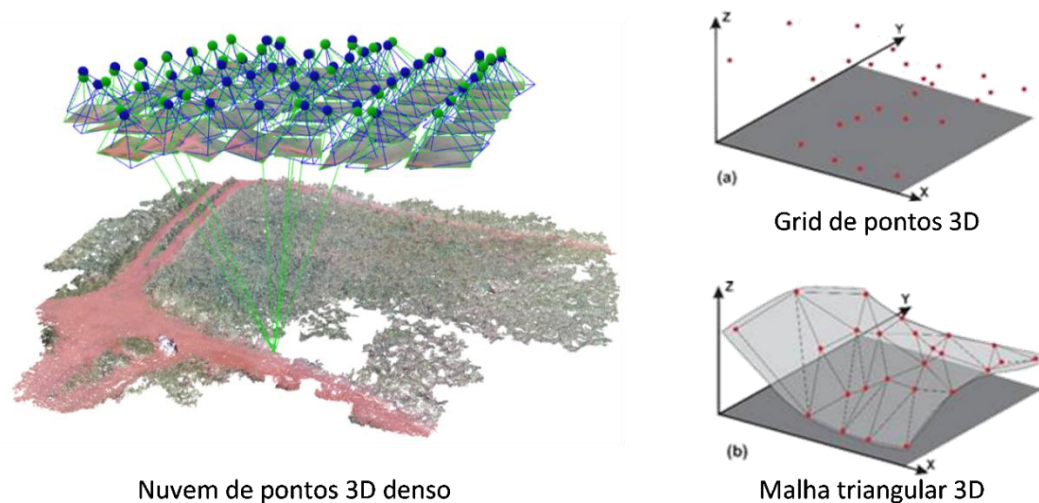
Na Figura 4.19, vemos múltiplas visualizações de um ponto 3D da nuvem de pontos esparsa, ou seja, cada ponto 3D no terreno é coberto por várias imagens sobrepostas.



**Figura 4.19** – Múltiplas visualizações de um ponto 3D nas imagens sobrepostas, mostrando feixes de raios e os *keypoints* correspondentes.

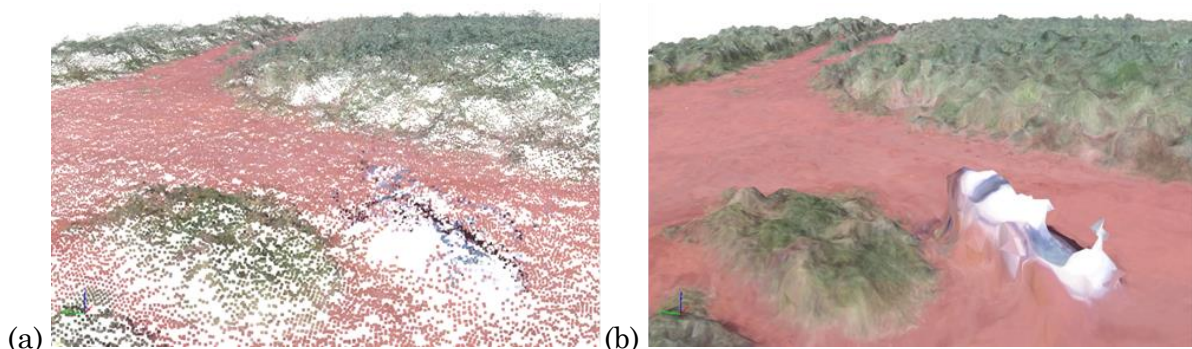
### 4.3.2 Etapa 2 – Nuvem de pontos densificada e malha 3D

A aerotriangulação é o processo de se determinar as coordenadas de pontos no terreno (espaço-objeto) dos pontos fotogramétricos correspondentes (*keypoints* homólogos) no espaço-imagem, usando as equações de colinearidade destes pontos [BRITO e COELHO 2012]. Após a Etapa 1 de orientação interna e externa das imagens, o algoritmo de visão estéreo múltipla (*Multiple View Stereo – MVS*) [FURUKAWA e HERNÁNDEZ 2015] realiza a aerotriangulação para gerar uma nuvem de pontos 3D densificada a partir de qualquer ponto das imagens estereoscópicas. Neste caso, mais pontos 3D no terreno são criados para gerar uma nuvem de pontos densa, com o objetivo de obter um *grid* regular de pontos [WOLF *et al.* 2014]. Em seguida, uma malha triangular 3D é criada a partir da nuvem de pontos densificada<sup>6</sup>, usando métodos de triangulação de Delaunay [BRITO e COELHO 2012], como ilustrado na Figura 4.20.



**Figura 4.20** – Densificação da nuvem de pontos 3D e triangulação para geração da malha 3D para visualização do modelo 3D do terreno.

A Figura 4.21a mostra uma ampliação da nuvem de pontos densificada, ou seja, um conjunto de pontos 3D, onde a posição X, Y, Z e as informações de cor são armazenadas para cada ponto da nuvem. A Figura 4.21b mostra uma malha 3D texturizada, ou seja, uma representação da forma do modelo, que consiste em vértices, arestas, faces e textura das imagens projetadas. A malha permite a visualização do modelo 3D do terreno, mas não tem precisão para realizar medições de tamanho, distância, área, volume, etc.



**Figura 4.21** – Nuvem de pontos 3D densa e malha texturizada 3D sem precisão métrica.

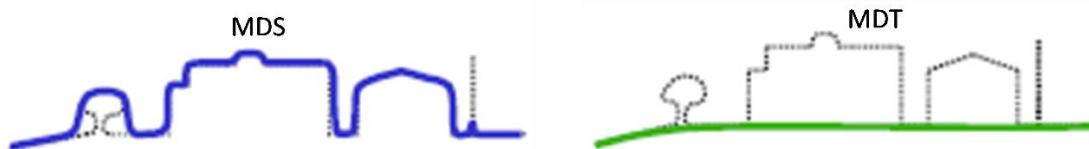
<sup>6</sup>A nuvem de pontos densa gera malhas mais detalhadas do que a nuvem esparsa. Os edifícios em áreas urbanas são difíceis de modelar sem uma nuvem de pontos densa. No entanto, os resultados são bons para áreas planas, como terras agrícolas [OPENDRONEMAP].



### 4.3.3 Etapa 3 – Modelos digitais de elevação

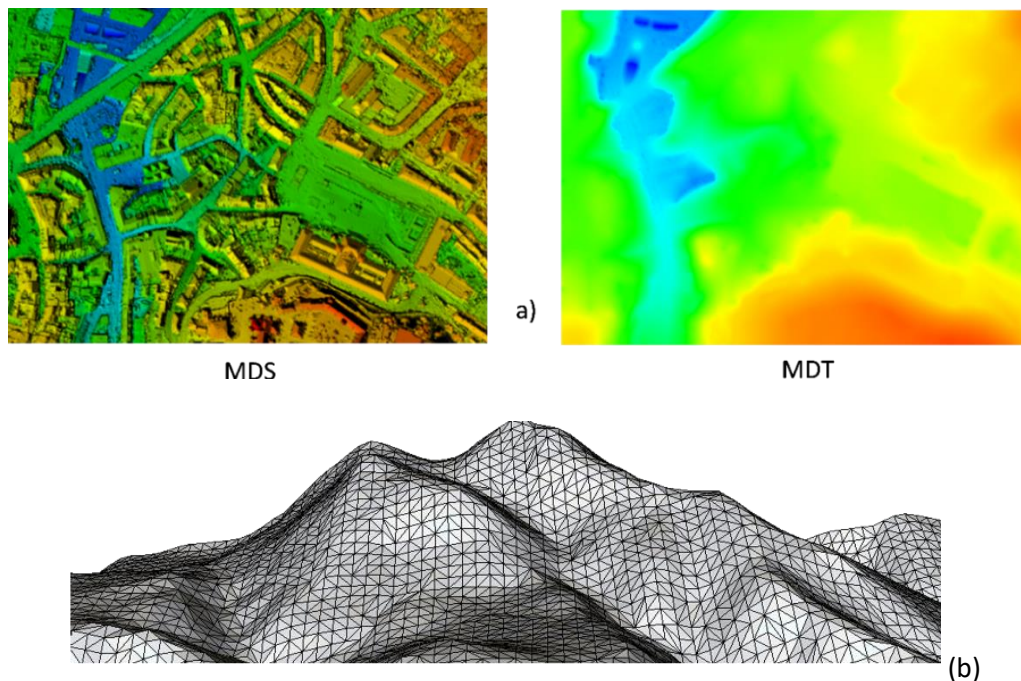
A Etapa 3 gera Modelos Digitais de Elevação (MDE) a partir da nuvem de pontos densificada [BRITO e COELHO 2012], representando o terreno por uma malha 3D ou imagem *raster* em três dimensões. Existem dois modelos derivados do MDE – Modelo Digital de Superfície (MDS) e Modelo Digital de Terreno (MDT) – que representam a elevação da superfície e do terreno, respectivamente, como ilustra a Figura 4.22.

O modelo digital de superfície representa o modelo tridimensional com a elevação da superfície, contemplando todos os objetos acima do solo, como vegetação, prédios, carros, pessoas, etc. O MDS permite o cálculo de volumes e a geração de ortomosaicos. Após filtragem do MDS, por exemplo, usando uma série de operações morfológicas de abertura (*Simple Morphological Filter* – SMRF) [SMRF; PINGEL *et al.* 2013], o modelo digital de terreno consiste na representação apenas do relevo, sem os objetos naturais e artificiais acima do solo. A partir do MDT é possível extrair curvas de nível, que são linhas imaginárias que possuem a mesma altitude, representando a declividade do terreno.



**Figura 4.22** – Modelo digital de superfície (MDS) e de terreno (MDT).

A Figura 4.23a mostra MDEs representados em formato *raster* GeoTIFF, onde cada valor de intensidade de *pixel* representa uma elevação visualizada em cores. Ao contrário da malha texturizada 3D que permite modelar qualquer objeto tridimensional, no MDS representado em formato de malha, o *grid* da nuvem de pontos tem apenas um valor Z de elevação (ponto mais alto) para cada posição (X, Y) do terreno.



**Figura 4.23** – Modelos digitais de elevação em formato *raster* GeoTIFF (a); e malha de pontos 3D gerada por triangulação de Delaunay [WOLF *et al.* 2014] (b).



#### 4.3.4 Ortorretificação de imagens aéreas (ortofotos)

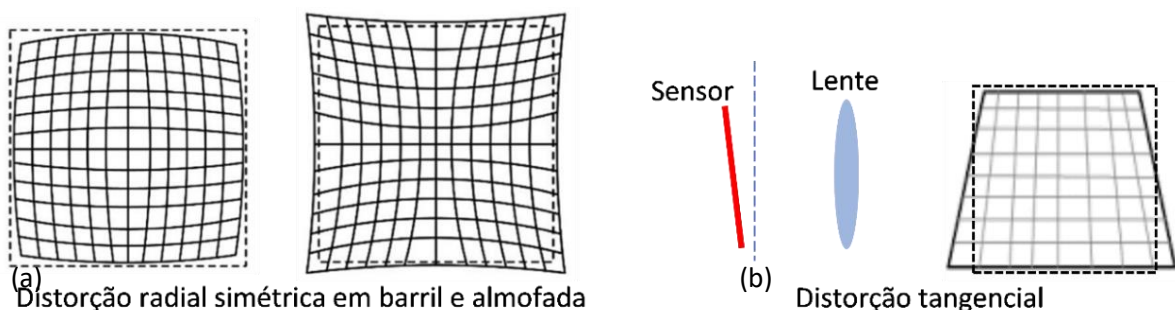
Durante o processo de aquisição, as imagens aéreas apresentam distorções causadas pelo tipo de lente e obturador, distorções de perspectiva devido à orientação do VANT, diferenças de escala devido às variações de altura de voo, pontos de vista diferentes devido ao deslocamento do VANT, distorções devido ao relevo do terreno e altura dos objetos no solo, além de alterações radiométricas (brilho e cor) devido às mudanças na iluminação durante o voo. Conseqüentemente, há necessidade de correção das distorções (retificação) das imagens para geração do mosaico.

Na Etapa 4, o software aerofotogramétrico realiza uma orrorretificação das imagens aéreas para geração do mosaico com auxílio do MDS [STRECHA *et al.* 2018]. Imagens orrorretificadas são comumente denominadas ortofotos, ou seja, imagens sem distorções devido à geometria de imageamento e relevo, que permitem extração de medidas [JENSEN 2009]. A seguir, discutimos as características de algumas destas distorções.

##### 4.3.4.1 Distorções da lente

Distorções geométricas da imagem ocorrem devido ao formato e posição das lentes que compõem o sistema da câmera. As distorções da lente não degradam a qualidade da imagem, mas deterioram a qualidade geométrica (ou precisão posicional) da imagem. As distorções causadas pelas lentes tendem a ser maiores nos sensores não fotogramétricos de pequeno formato usados em VANTs.

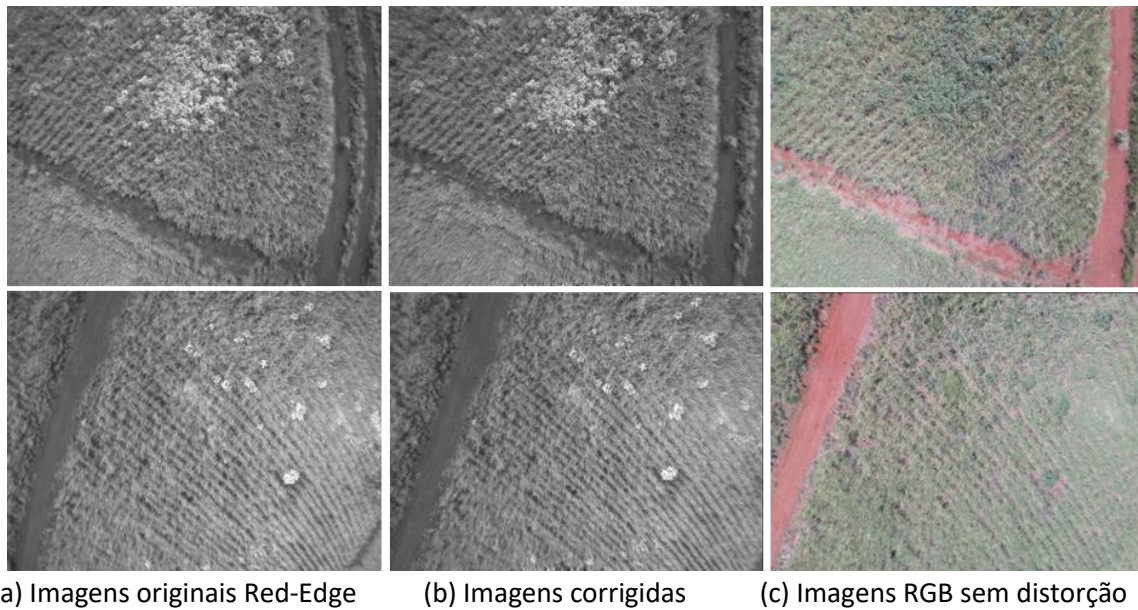
As distorções da lente são compostas pelas componentes radial simétrica e descentrada [BRITO e COELHO 2012; WOLF *et al.* 2014]. As distorções radiais simétricas podem ser de dois tipos: *barrel distortion* ou *pincushion distortion* (Figura 4.24a). A distorção radial em barril é um efeito associado a lentes grande-angulares, principalmente lentes olho-de-peixe (*fisheye lens*) que apresentam maior distorção. Esta distorção é maior quanto mais distante do centro da imagem. A distorção descentrada ou tangencial (Figura 4.24b) é gerada pela impossibilidade do fabricante em alinhar os eixos ópticos das lentes em relação ao sensor, resultando em um deslocamento na imagem. Tanto a distorção radial quanto a distorção tangencial são modeladas por equações matemáticas, cujos coeficientes de distorção específicos de cada câmera são fornecidos pelos fabricantes [WOLF *et al.* 2014]. Desta forma, os softwares de fotogrametria modernos, como PIX4Dmapper, corrigem essas distorções automaticamente.



**Figura 4.24** – Distorção radial simétrica e distorção tangencial [WOLF *et al.* 2014].

As distorções radial e tangencial da lente perspectiva RGB da câmera Sequoia não são perceptíveis visualmente. Entretanto, as distorções radiais em barril nas imagens originais multiespectrais, causadas pelo maior ângulo de visão (FoV – *Field of View*) da lente grande angular do tipo olho-de-peixe usadas nos sensores multiespectrais, são

visualmente perceptíveis na [Figura 4.25a](#), em comparação com as imagens corrigidas sem distorção geradas na Etapa 1.

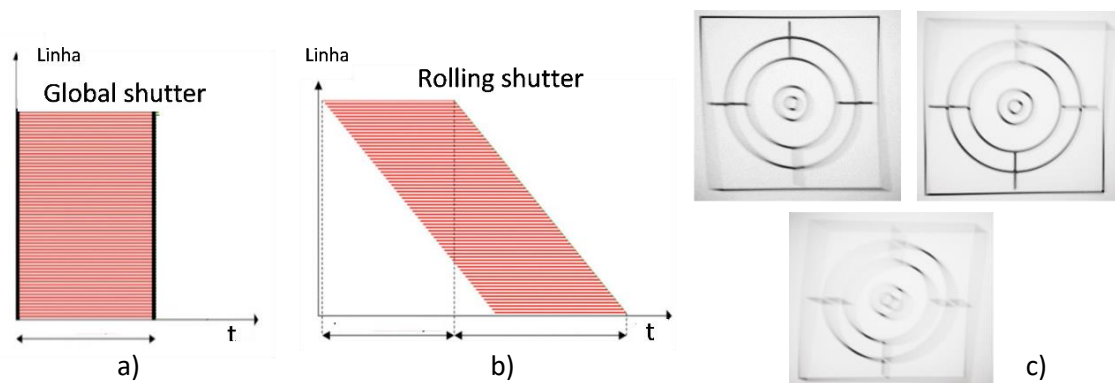


**Figura 4.25** – Imagem multiespectrais com maior distorção radial da lente olho-de-peixe.

#### 4.3.4.2 Distorções por obturador eletrônico

No obturador global (*global shutter*), usado em câmeras profissionais equipadas com sensores CCD, todas as linhas da imagem são integradas ao mesmo tempo, capturando o quadro inteiro de uma só vez ([Figura 4.26a](#)). Os sensores CMOS com obturador de rolamento (*rolling shutter*) capturam a imagem linha por linha ([Figura 4.26b](#)) [MEINGAST *et. al.* 2005]. Esse pequeno atraso entre a leitura da primeira e última linha pode alterar perspectivas na imagem, criando distorções indesejadas onde linhas retas podem parecer inclinadas ou curvas, se a câmera ou objeto se desloca rapidamente na cena. Isto causa um efeito *rolling shutter* significativo (em inglês, *jelly effect*) ao voar com maior velocidade do VANT ou com vibrações das hélices, introduzindo distorções ([Figura 4.26c](#)) que reduzem a confiabilidade da correspondência entre as imagens [HOLLER *et. al.* 2022].

A câmera não fotogramétrica Sequoia tem um obturador eletrônico do tipo *rolling shutter* associado ao sensor RGB e *global shutter* nos sensores multiespectrais. O efeito da distorção *rolling shutter* é modelado matematicamente pelo software aerofotogramétrico de VANT. O PIX4Dmapper corrige estas distorções automaticamente na Etapa 1.



**Figura 4.26** – Distorção causada pelo rolling shutter da câmera em movimento.

#### 4.3.4.3 Distorções de inclinação da câmera

Em relação à orientação do eixo ótico da câmera, as fotografias aéreas são classificadas em vertical (eixo ótico da câmera até  $\pm 3^\circ$ ) ou oblíqua (inclinação maior que  $3^\circ$ ) [WOLF *et al.* 2014]. Para evitar distorções de projeção perspectiva em projetos de mapeamento aéreo, as imagens são capturadas orientando-se o eixo ótico verticalmente (perpendicular ao solo), de modo que a câmera esteja apontando diretamente para baixo (posição nadir). No entanto, VANTs leves são bastante vulneráveis à influência do vento. Uma forma de diminuir este problema é adicionar um sistema de estabilização da câmera, chamado *gimbal*. Quando não usado (como no nosso caso), o resultado é um voo instável, conforme mostrado na Figura 4.27.

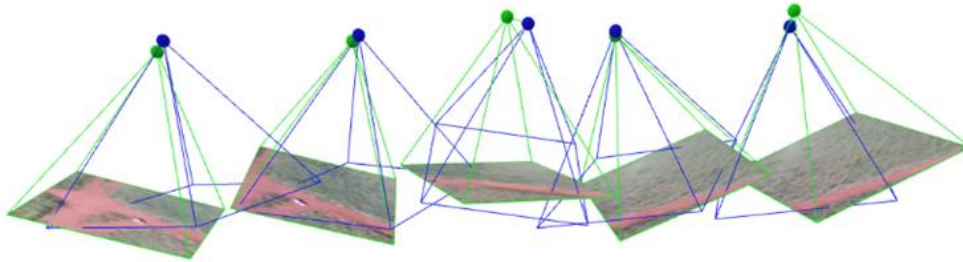


Figura 4.27 – Instabilidade do VANT sem estabilizador de câmera (*gimbal*).

Os movimentos nas direções X e Y (*roll*, *pitch*) também afetam a sobreposição lateral e frontal, respectivamente, ao longo da linha de voo, como mostra a Figura 4.28.

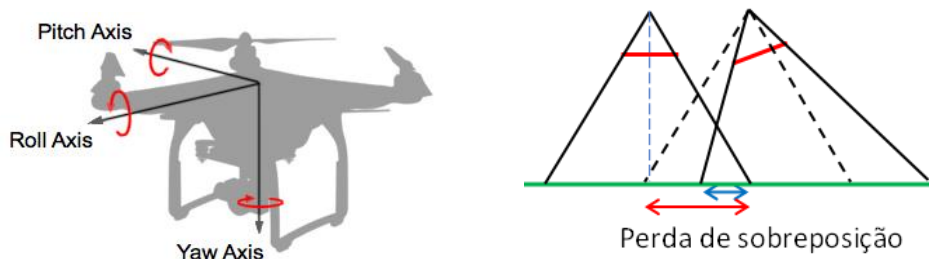


Figura 4.28 – Perda de sobreposição frontal e/ou lateral devido à inclinação da câmera. Adaptado de [WOLF *et al.* 2014].

Além disso, a posição *off-nadir* causa distorções de perspectiva a partir do centro de projeção CP, que devem ser corrigidas pelo software aerofotogramétrico para projetar as imagens sem distorções no processo de geração do ortomosaico (Figura 4.29).

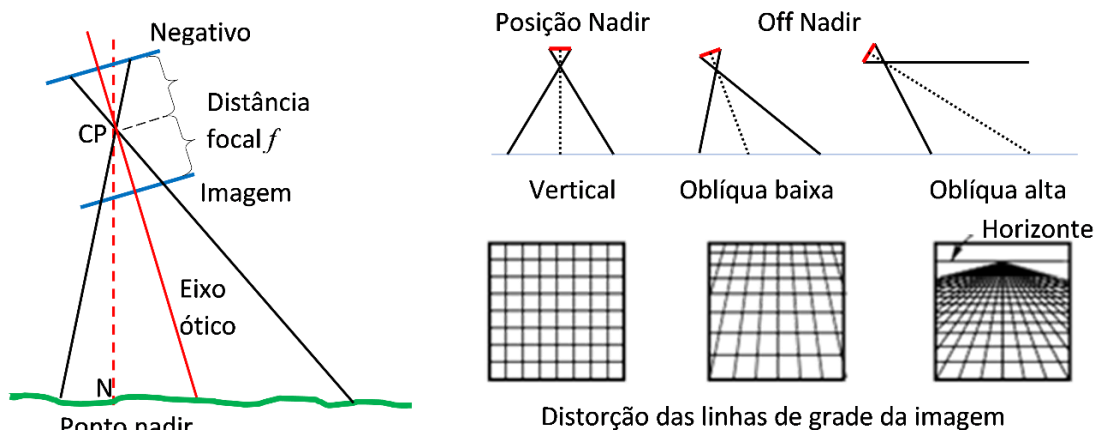
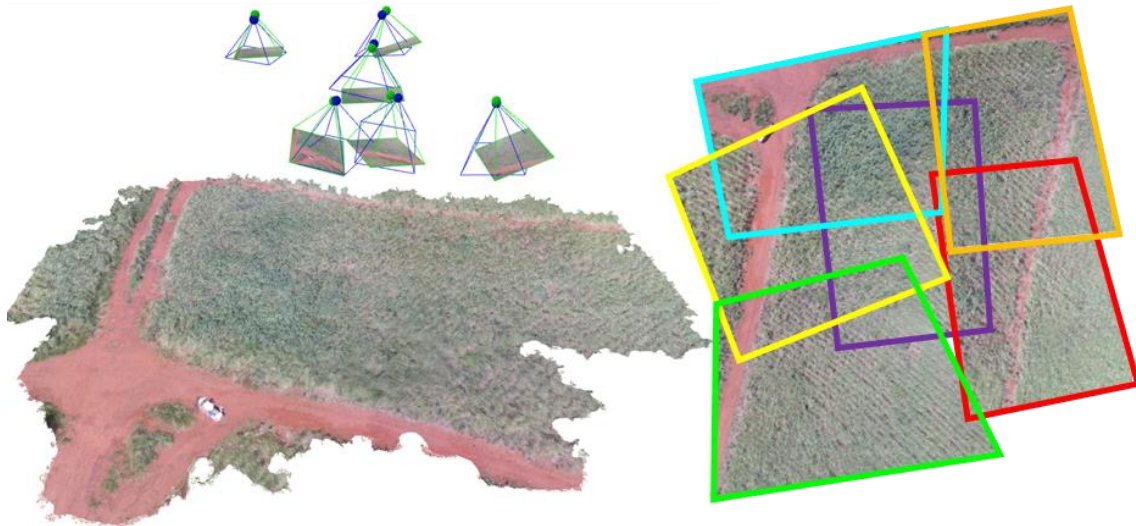


Figura 4.29 – Distorção perspectiva de acordo com a inclinação da câmera [WOLF *et al.* 2014].

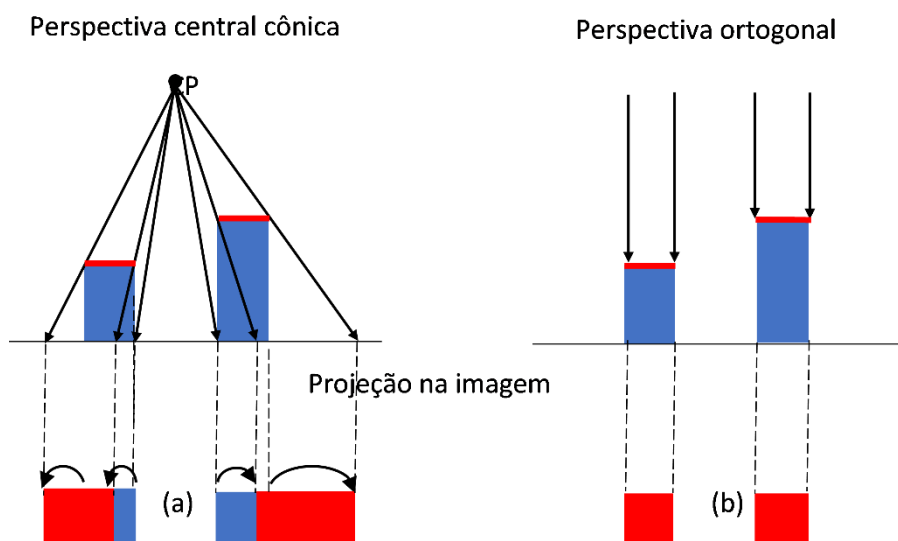
Para ilustrar este problema, podemos observar na [Figura 4.30](#) que algumas imagens originais apresentam distorção de perspectiva devido ao ângulo de inclinação da câmera. As transformações geométricas afim das imagens foram geradas manualmente sobre o mosaico 2D, apenas para efeitos de ilustração.



**Figura 4.30** – *Imagens originais com distorções de perspectiva devido à inclinação da câmera e transformação geométrica afim das imagens sobre o ortomosaico 2D.*

#### 4.3.4.4 Distorções de perspectiva cônica

A [Figura 4.31](#) mostra dois tipos de projeção das imagens – perspectiva central cônica e perspectiva ortogonal. A ortorretificação de uma imagem tem como finalidade corrigir as distorções de perspectiva central cônica, colocando a mesma na posição correta, como se tivesse sido observada na vertical com uma projeção de perspectiva ortogonal [[BRITO e COELHO 2012](#)].

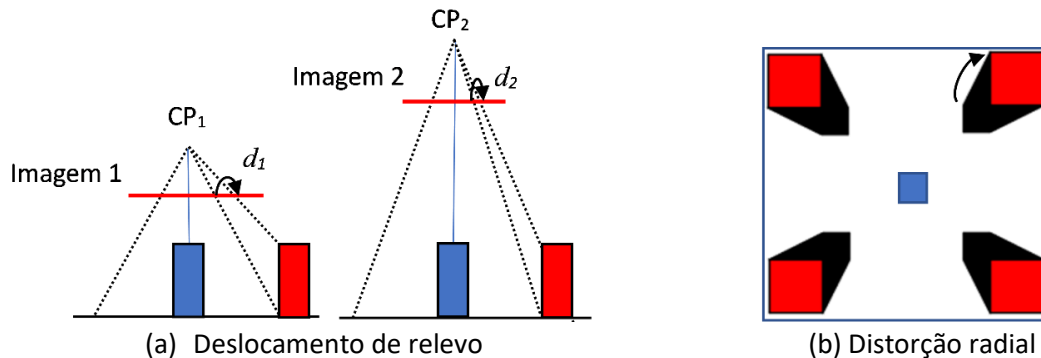


**Figura 4.31** – *Tipos de projeção de imagens: (a) projeção perspectiva central cônica a partir de um centro de projeção CP e (b) projeção perspectiva ortogonal a partir do infinito. Baseado em [[BRITO e COELHO 2012](#)].*



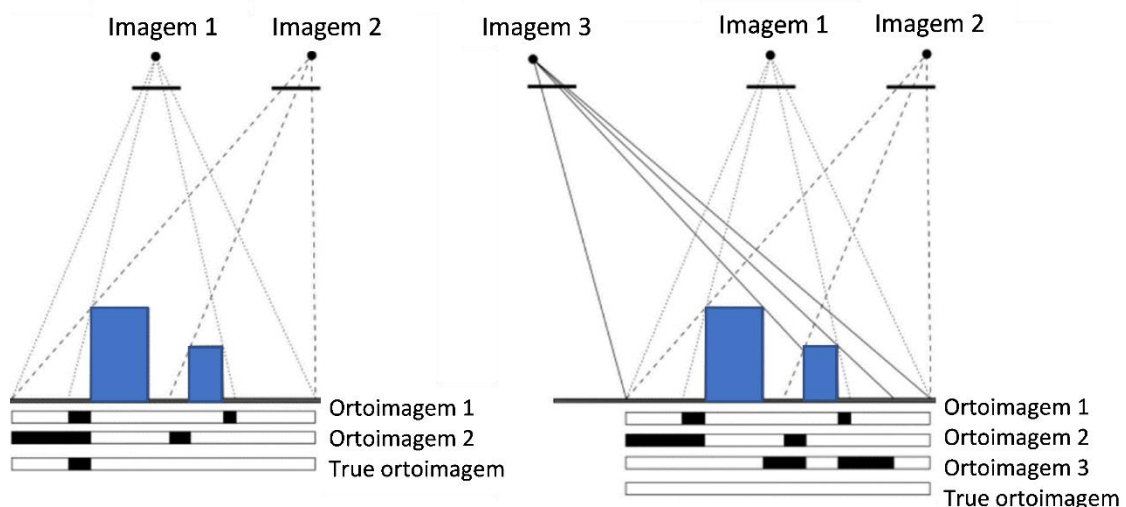
A projeção cônica gera uma deformação de perspectiva, de forma que, à medida que se afasta do centro da imagem, objetos altos aparecem inclinados na direção radial, permitindo a visualização das laterais dos objetos. Desta forma, o topo dos objetos é deslocado no plano da imagem, como mostra a [Figura 4.31a](#) e [Figura 4.32b](#).

O efeito de deslocamento do “relevo” varia de acordo com a distância do ponto central da imagem, altura do objeto e altura da plataforma aérea. Esta distorção é radial a partir do ponto central da imagem aérea. Quanto mais afastado um objeto do centro da imagem, maior o deslocamento. Quanto maior a altura de voo, menor o deslocamento ( $d_2 < d_1$ ) do “relevo” ([Figura 4.32a](#)). Além disso, observando na [Figura 4.32b](#), as imagens aéreas apresentam uma mudança perceptível de escala entre o topo e a base de objetos.



**Figura 4.32** – (a) Deslocamento do “relevo” causado pela projeção cônica em duas alturas de voo; (b) Distorção radial com deslocamento a partir do centro da imagem. Baseado em [WOLF *et al.* 2014].

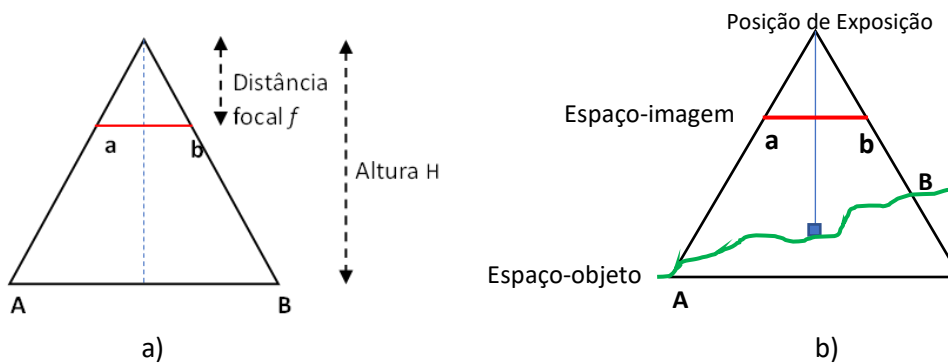
As projeções perspectivas cônicas causam uma oclusão na imagem projetada devido à altura do objeto ou altura de voo em determinado ângulo de visão. A [Figura 4.33a](#) mostra que a projeção perspectiva de um objeto mais alto gera uma maior área de oclusão na imagem projetada, ocultando informações que precisam ser obtidas a partir de outras imagens aéreas com outros pontos de vista. Com isso, há necessidade de se obter mais ortofotos de imagens adjacentes para compor a imagem ortorretificada sem oclusões ([Figura 4.33b](#)), chamada ortofoto verdadeira (*true orthoimage*) [STRECHA *et al.* 2018].



**Figura 4.33** – Geração de uma ortofotografia verdadeira. (a) Duas imagens geram oclusões na ortofotografia verdadeira. (b) Três imagens contribuem com a ortofotografia verdadeira sem oclusões devido aos diferentes ângulos de visão.

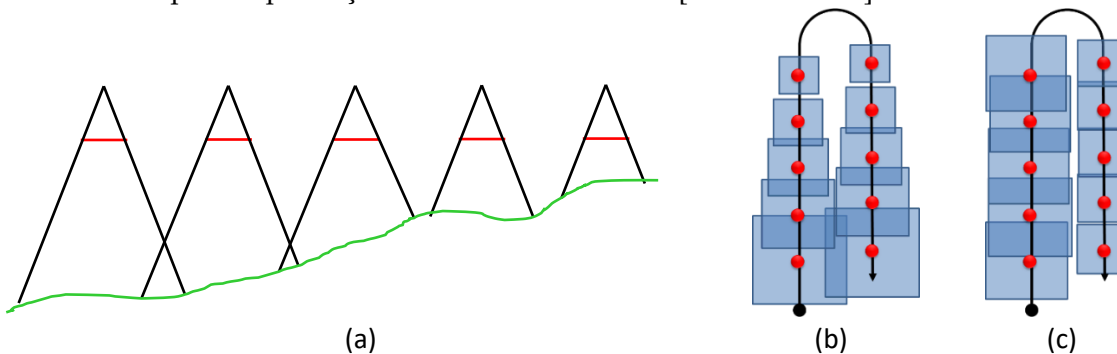
**4.3.4.5 Distorções de relevo**

A escala de uma fotografia vertical sobre um terreno plano pode ser expressa pela razão entre a distância focal da lente e a altura de voo ( $E = f / H$ ), ou entre os tamanhos dos objetos na imagem e no terreno ( $E = ab/AB$ ), como mostra a [Figura 4.34a](#) [BRITO e COELHO 2012; WOLF *et al.* 2014; JENSEN 2009]. Por outro lado, há um número infinito de diferentes escalas presentes nas imagens aéreas adquiridas sobre um terreno de relevo acidentado ([Figura 4.34b](#)). Se a elevação topográfica diminui dentro de certa área da fotografia aérea, então essa área terá uma escala menor, porque o solo está mais distante da câmera. De forma inversa, se o relevo, tal como uma montanha ou uma edificação elevada, estiver acima da média da elevação do terreno local, então a escala nessa área da fotografia será maior, uma vez que o solo está mais próximo da câmera. Desta forma, objetos mais próximos da câmera parecem maiores, como o topo de edifícios ou montanha em relação à sua base [[JENSEN 2009](#)].



**Figura 4.34** – Escala de uma imagem sobre um terreno plano (a); Variação de escala de objetos de uma imagem aérea devido ao relevo irregular (b). Adaptado de [[WOLF \*et al.\* 2014](#); [JENSEN 2009](#)].

O mapeamento aéreo de um terreno acidentado também causa variações na resolução GSD entre as imagens e, conseqüentemente, variação de escala e tamanho dos objetos no mosaico. A [Figura 4.35a](#) mostra uma situação onde a altitude de voo acima do nível do mar permanece constante, mas ocorre a variação de escala devido à variação da altura de voo acima do nível do solo (distância do objeto) causada pelo relevo irregular. Isto significa que, se a distância focal for mantida constante, as imagens dos objetos serão maiores conforme se diminui a altura de voo. Além disso, a sobreposição frontal ([Figura 4.35b](#)) ou lateral ([Figura 4.35c](#)) e a área de cobertura diminuem conforme a elevação do terreno aumenta, prejudicando a obtenção de fotografias estereoscópicas necessárias para a produção de MDEs e ortofotos [[JENSEN 2009](#)].



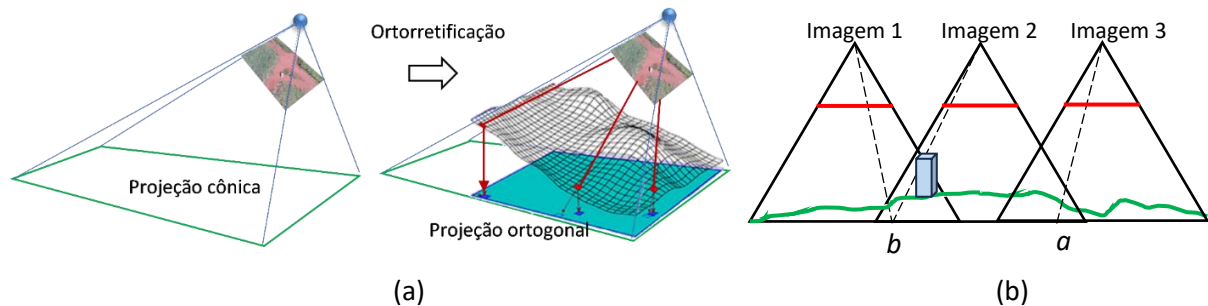
**Figura 4.35** – Variação de escala devido ao relevo irregular (a); redução da sobreposição frontal (b) e lateral (c). Adaptado de [[WOLF \*et al.\* 2014](#)].



#### 4.3.4.6 Ortorretificação de imagens do mosaico

Como vimos, além de corrigir as distorções internas da câmera (distorção da lente, etc.), o processo de ortorretificação das imagens aéreas deve corrigir também as distorções externas na geometria das imagens, causadas pela orientação da câmera e relevo [BRITO e COELHO 2012]. Assim, a projeção perspectiva cônica da imagem é transformada em uma projeção ortogonal sobre o MDS, onde todos os pontos da imagem parecem ser vistos de cima a partir de um ponto de visão no infinito (Figura 4.36a). Dessa forma, é possível eliminar distorções de projeção perspectiva cônica decorrentes da inclinação da câmera. No entanto, uma ortorretificação verdadeira é necessária para remover distorções de deslocamento de relevo no mosaico.

Na Figura 4.36b, observam-se três fotografias aéreas e um MDE sobre a área de estudo. O valor do *pixel a* é interpolado a partir da imagem mais próxima ao nadir (diretamente acima), que possui a melhor visualização do terreno na localização *a* (no caso, imagem 3). A visualização do terreno para o *pixel b* é ocultada pelo edifício na imagem 2, então a imagem 1 é selecionada automaticamente, visando obter o valor de *pixel* correto para o *pixel b*. A aplicação desse algoritmo resulta no ortomosaico verdadeiro [JENSEN 2009].



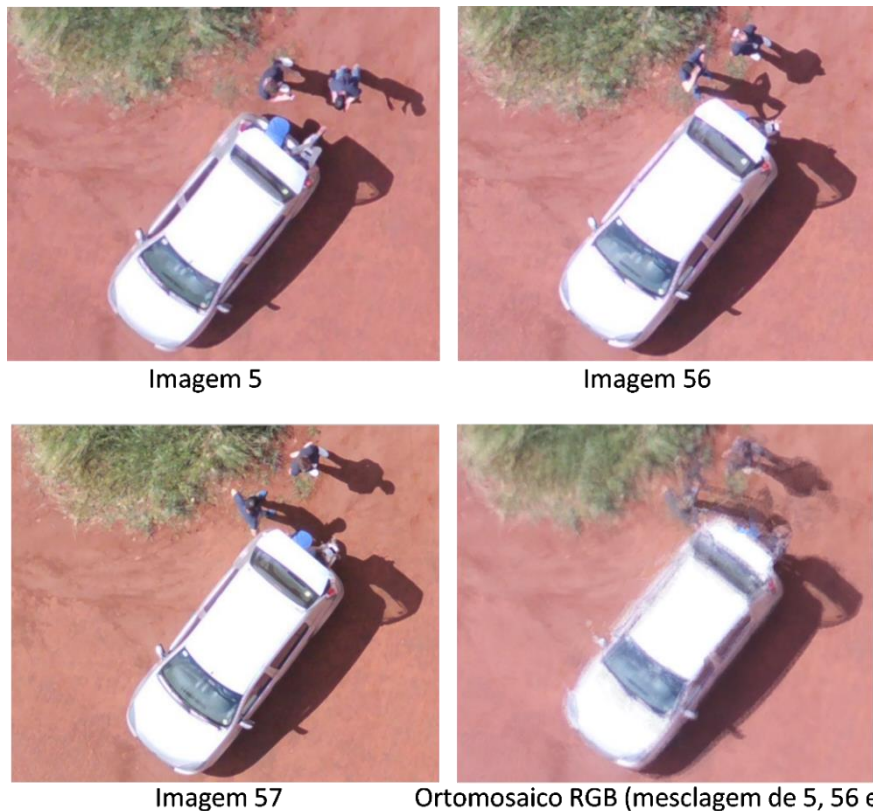
**Figura 4.36** – *Projeção perspectiva central cônica (retângulo verde vazio) e projeção ortogonal (retângulo cheio) para ortorretificação da imagem sobre um relevo (a); projeção ortogonal sobre MDE a partir de múltiplas imagens para gerar ortomosaico verdadeiro (b). Adaptado de [JENSEN 2009].*

#### 4.3.5 Etapa 4 – Ortomosaico 2D

Para geração de ortomosaicos 2D na Etapa 4, a ortorretificação projeta os *pixels* das imagens aéreas sobre um modelo digital de superfície, usando uma projeção ortogonal (ou planar, no caso de áreas de menor interesse, para diminuir o tempo de computação). Todas as distorções de perspectiva devido à obliquidade da câmera e relevo irregular do terreno são removidas para que as características dos objetos e do terreno sejam mostradas em suas posições geográficas corretas. Desta forma, ortomosaicos combinam as características de uma imagem aérea de alta resolução com as qualidades geométricas de um mapa cartográfico, que permitem realizar medidas de distâncias, posições, ângulos, etc.

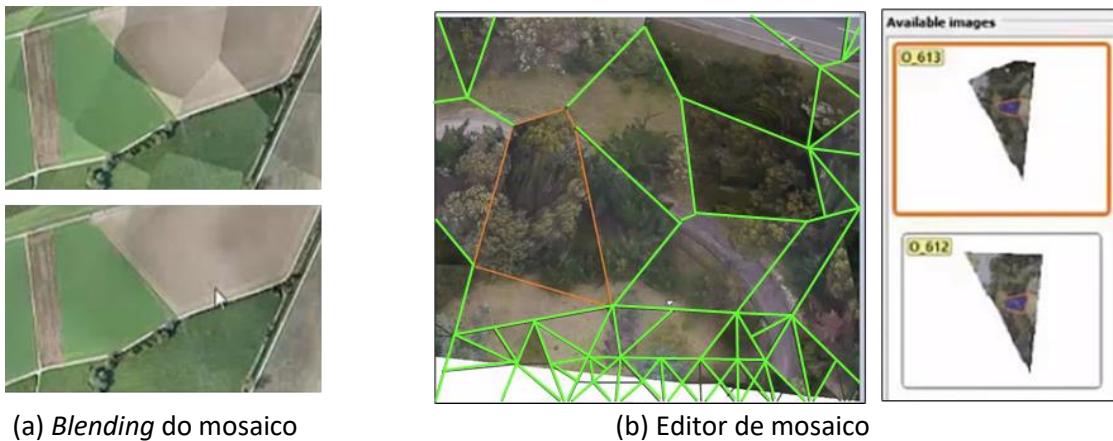
Um ortomosaico verdadeiro não é apenas uma projeção de *pixels* das imagens aéreas no MDS, mas um mosaico de várias partes de ortofotos. Assim, as oclusões são preenchidas com partes de imagens adjacentes que apresentam menor distorção perspectiva, sendo que o valor de cada *pixel* é obtido como uma média ponderada dos *pixels* correspondentes nas imagens. Áreas com muitos objetos verticais ou com relevo acentuado podem exigir maior quantidade de imagens sobrepostas [STRECHA *et al.* 2018].

Na [Figura 4.37](#), vemos uma parte do mosaico gerado pela ortorretificação de três imagens originais com mesclagem de *pixels* e degradação da qualidade da imagem.



**Figura 4.37** – Detalhes das imagens originais e mesclagem dos *pixels* para gerar uma parte do ortomosaico RGB.

Além disso, as partes de ortofotos que formam o mosaico apresentam diferenças de radiometria devido às variações de iluminação durante o mapeamento aéreo ([Figura 4.38a](#)). Portanto, uma homogeneização da radiometria usando técnicas de *blending* é necessária para remover as costuras [[BROWN e LOWE 2007](#)]. Se for necessário melhorar o aspecto visual em determinadas partes do mosaico, o editor de mosaicos do PIX4Dmapper permite selecionar o tipo de projeção (planar ou ortogonal) e a imagem a ser projetada em cada polígono, removendo artefatos durante a projeção ([Figura 4.38b](#)).



**Figura 4.38** – (a) *Blending* do mosaico; (b) Editor de mosaico do PIX4Dmapper que permite seleção da imagem a ser projetada no polígono selecionado [[PIX4DMAPPER](#)].

Resumindo, uma ortofotografia, também conhecida como ortofoto ou ortoimagem é uma imagem aérea que foi corrigida geometricamente (ortorretificada). Ortomosaicos de imagens aéreas são comumente denominados mosaico de ortofotos. O arquivo do ortomosaico em formato GeoTIFF (com mistura de cores de várias ortofotos) é georreferenciado, ou seja, cada *pixel* da imagem possui suas coordenadas geográficas conhecidas, possibilitando a realização de medidas no sistema de coordenadas de terreno (centímetros ou metros). No entanto, não é possível mensurar a altura dos objetos no mosaico 2D, sendo, para isso, necessária a utilização do MDS.

#### 4.3.6 Etapa 4 – Mapas de reflectância

No projeto PIX4Dmapper multiespectral, quando as entradas são imagens monocromáticas multiespectrais de todas as bandas espectrais, em formato GeoTIFF sem compressão, com valores de *pixels* entre 0 e 65.535 (inteiro sem sinal de 16 bits), um ortomosaico monocromático é gerado para cada banda *Green*, *Red*, *Red-Edge* e *NIR* no mesmo formato de dados. Entretanto, um mapa de reflectância com calibração radiométrica pode ser gerado para cada banda espectral, em formato GeoTIFF, onde cada *pixel* indica um valor de reflectância do objeto, com valores calibrados radiometricamente entre 0 e 1 (ponto flutuante de 32 bits). O mapa de reflectância, da mesma forma que o ortomosaico multiespectral, é gerado na Etapa 4 pela ortorretificação das imagens aéreas multiespectrais no MDS [STRECHA *et al.* 2018].

No PIX4Dmapper, se gerarmos um ortomosaico monocromático multiespectral, um balanceamento é aplicado para ajustar a intensidade de tons de cinza das imagens em diferentes partes do mosaico, produzindo um resultado visualmente uniforme. No entanto, nenhum balanceamento é aplicado no mapa de reflectância e os pesos de ponderação da mistura de *pixels* são calculados de forma diferente do ortomosaico, para que o valor de cada *pixel* indique a reflectância real do objeto. Antes da execução da Etapa 4 para geração dos mapas de reflectância, uma calibração radiométrica das imagens multiespectrais deve ser realizada.

##### 4.3.6.1 Calibração radiométrica de imagens multiespectrais

Normalmente, dados adquiridos pela plataforma VANT são afetados pelas características do sensor, condições de iluminação, alinhamento geométrico e condições atmosféricas. Conseqüentemente, os números digitais (DNs) não são verdadeiros representantes da reflectância da superfície. No entanto, para monitorar as terras agrícolas com imagens multiespectrais adquiridas em diferentes altitudes de voo, instantes de tempo e condições meteorológicas, elas devem ser calibradas radiometricamente, levando em conta as condições atmosféricas, solares e topográficas, bem como as características da câmera [GUO *et al.* 2019].

A calibração radiométrica tem como objetivo converter os números digitais da imagem para gerar imagens multiespectrais com reflectância unitária, adequando os dados para uso direto no sensoriamento remoto quantitativo. As imagens calibradas radiometricamente podem ser usadas para calcular índices de vegetação, que são úteis no monitoramento da vegetação [GUO *et al.* 2019].

O valor do *pixel* da imagem depende de muitos fatores, que precisam ser corrigidos para obter uma medida radiometricamente confiável da reflectância do terreno. Esses fatores incluem:

- **Configurações do sensor:** velocidade do obturador, ISO, abertura;
- **Propriedades do sensor:** transmissão de luz na lente e digitalização no chip;
- **Condições da cena:** luz solar incidente, localização da câmera e orientação.

A correção radiométrica resolve esse problema corrigindo a reflectância das imagens multiespectrais, levando em consideração a iluminação da cena e a influência do sensor. Ao gerar o mapa de reflectância na Etapa 4, o PIX4Dmapper utiliza os valores de alguns parâmetros presentes no EXIF das imagens para corrigir (parcialmente) os seguintes fatores [PIX4DMAPPER]:

- **Sensor:** as correções levam em consideração as propriedades e configurações da câmera (vinheta, ISO, etc.). Esses parâmetros são obtidos dos metadados EXIF;
- **Irradiância solar:** as correções levam em consideração as informações fornecidas pelos sensores de irradiância solar (sensores de luz ou sensor solar do VANT), que captura a radiação recebida do sol. Estes sensores fornecem um registro das condições de luz durante o voo nas mesmas bandas espectrais captadas pelo sensor multiespectral. O PIX4Dmapper consegue normalizar as imagens captadas durante o voo e, assim, permite comparar imagens captadas em diferentes condições de iluminação;
- **Ângulo solar:** leva em consideração a direção do raio de sol incidente e sua projeção na cena e no sensor solar. Esta opção só deve ser selecionada para voos que foram realizados em condições de céu claro.

A câmera Sequoia pode voar sem o sensor de irradiância solar. No entanto, para produzir mapas de reflectância radiometricamente precisos, seria necessário um alvo de calibração radiométrica (também conhecido como alvo de reflectância). Estes alvos também podem ser usados para fazer uma calibração radiométrica adicional em condições de campo. O alvo de calibração radiométrica é um cartão de balanço de branco, como mostra a [Figura 4.39](#), que fornece as propriedades de reflectância do cartão em todo o espectro de luz capturado pela câmera multiespectral (ou seja, verde, vermelho, borda de vermelho, NIR e outras faixas dependendo na câmera).



**Figura 4.39** – Alvo de calibração radiométrica.

A utilização de um alvo de calibração radiométrica permite ao software calibrar e corrigir a reflectância das imagens de acordo com os valores fornecidos pelo alvo de calibração. A calibração leva em consideração as condições de iluminação na data, hora e local da captura da imagem, bem como algumas características do sensor. O alvo de calibração permite obter valores absolutos de reflectância e possibilita a comparação de dados provenientes de várias câmeras ou voos [HOLLER *et al.* 2022].

Nota: A câmera Sequoia também pode voar sem o sensor solar e sem alvos de reflectância, mas os resultados não serão calibrados radiometricamente e a comparação entre mapas não é recomendada. Portanto, recomenda-se usar um alvo de calibração ao gerar mapas de reflectância e mapas de índices [PIX4DMAPPER].

As imagens multiespectrais capturadas pelo CTBE foram calibradas radiometricamente com o sensor solar.

#### 4.3.7 Etapa adicional: Mapas de índices de vegetação

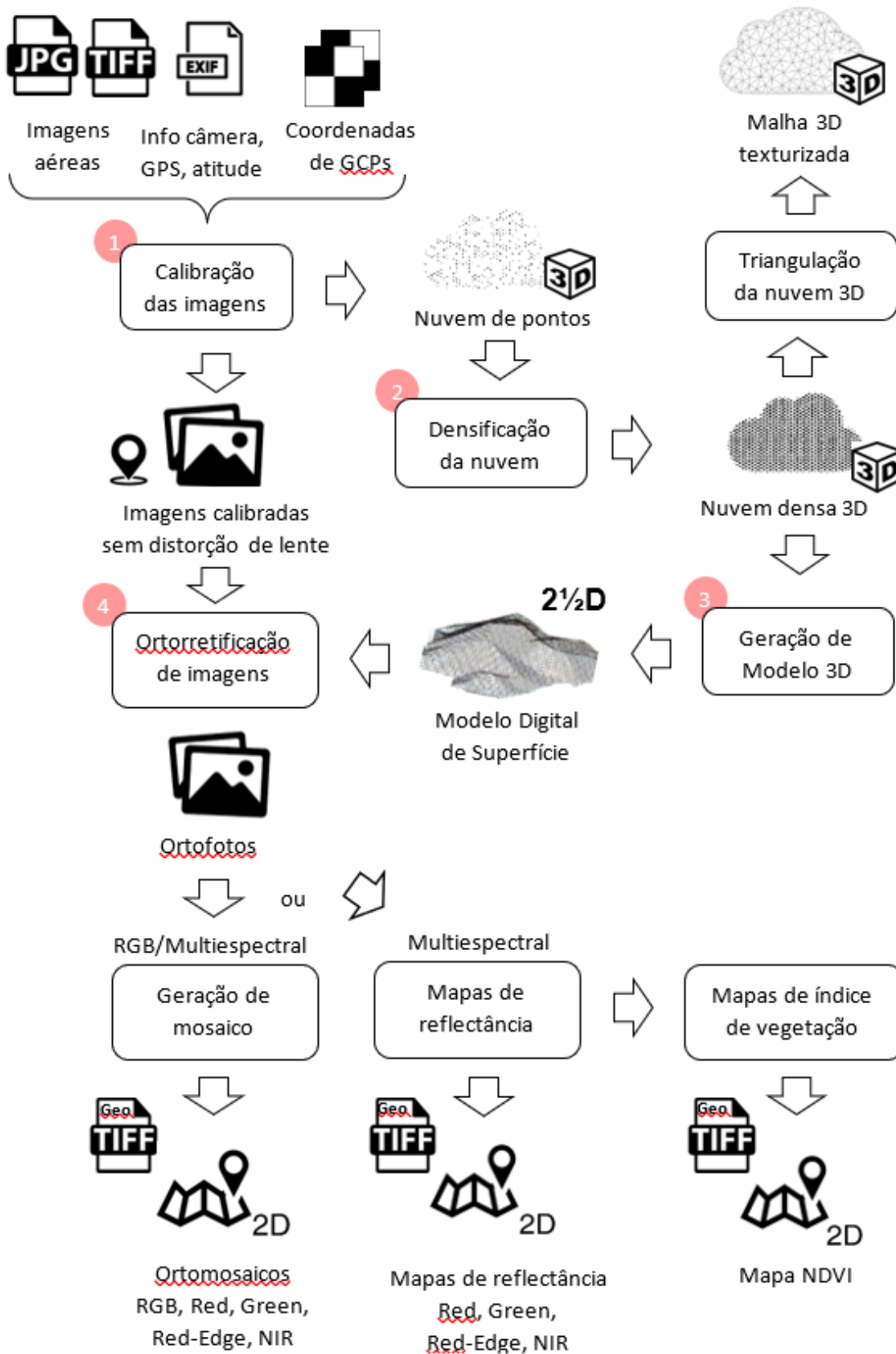
Os mapas de reflectância multiespectrais podem ser usados para gerar vários mapas de índice de vegetação na Etapa 4. O PIX4Dmapper gera o mapa de índice NDVI georreferenciado em GeoTIFF, com valores em ponto flutuante de 32 bits. Além disso, geramos outros mapas de índices de vegetação usando um sistema de informação geográfica (SIG) – QGIS [QGIS]. Este software livre e de código aberto permite a visualização, edição e análise de dados georreferenciados.

---

A [Figura 4.40](#) mostra o processo completo de processamento de imagens para geração de ortomosaicos RGB/multiespectrais, mapas de reflectância multiespectrais e mapas de índices de vegetação. Nas próximas seções, apresentamos as características dos produtos aerofotogramétricos de base cartográfica gerados pelo processamento das imagens do campo experimental de cana-de-açúcar.

Uma vez obtidos os produtos aerofotogramétricos, estes podem ser utilizados para uma análise nos sistemas de informação geográfica (SIG), como também em sistemas de aprendizado de máquina para tarefas de classificação e segmentação semântica de imagens, permitindo a geração de relatórios para tomadas de decisão na agricultura.



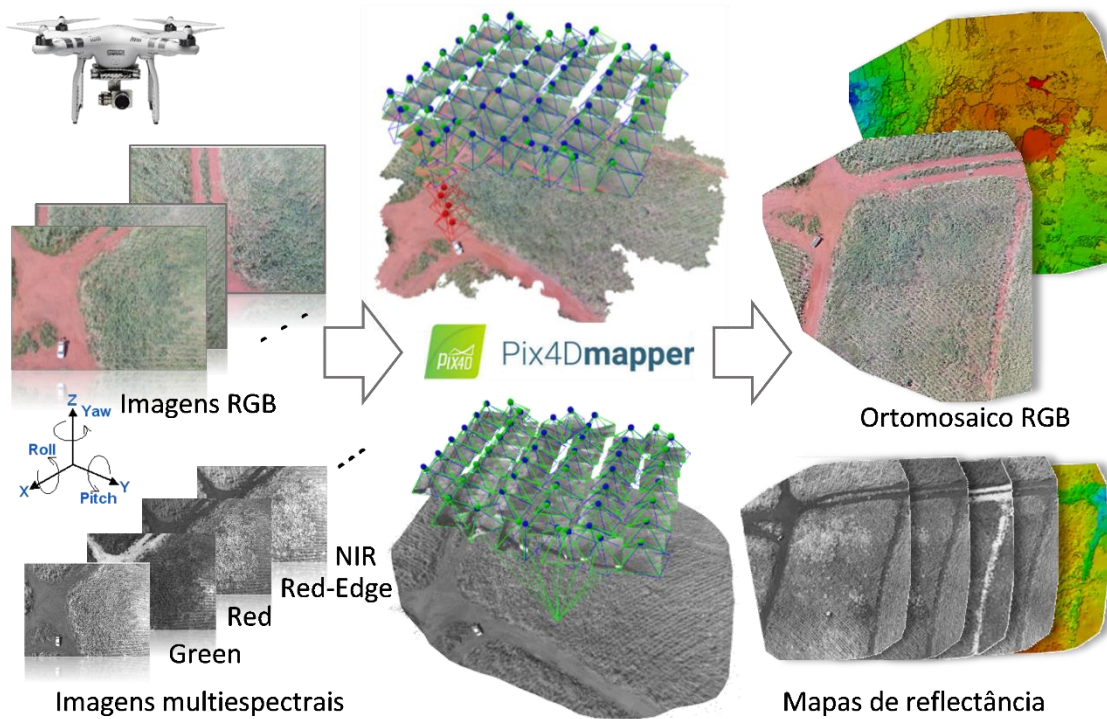


**Figura 4.40** – Esquema geral de processamento de imagens aéreas para geração da nuvem de pontos 3D, malha de pontos 3D, modelo de superfície 2½D, ortomosaicos 2D georreferenciados, mapas de reflectância 2D e mapas de índice de vegetação 2D.



#### 4.4 Processamento de imagens do campo experimental

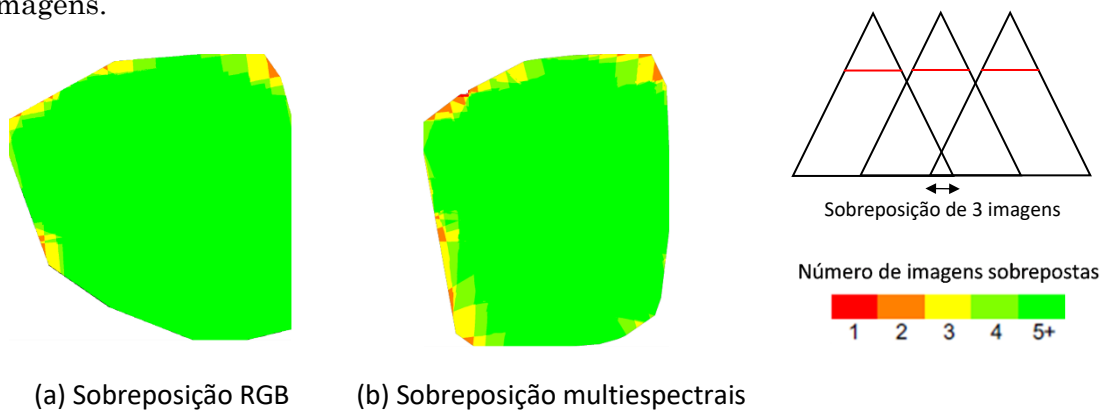
Com base nos conjuntos de imagens RGB e multiespectrais coletados sobre o campo experimental de cana-de-açúcar, listados na [Tabela 4.2](#), criamos dois projetos no PIX4Dmapper: MosaicoRGB e MapasReflectância ([Figura 4.41](#)).



**Figura 4.41** – Imagens de entrada e produtos de saída do PIX4Dmapper [PIX4DMAPPER].

##### 4.4.1 Processamento inicial e nuvem de pontos densificada

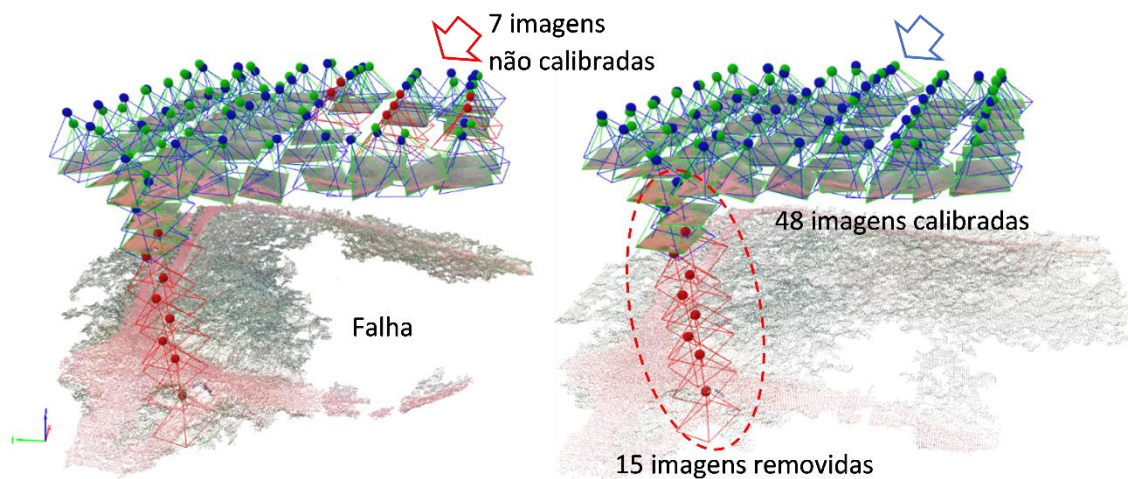
Em aplicações de agricultura, onde o terreno tem conteúdo visual homogêneo, é difícil extrair *keypoints* comuns entre as imagens. Para evitar este problema, deve-se usar alta sobreposição frontal e lateral na fase de aquisição de imagens. Na [Figura 4.42](#), as áreas vermelhas e amarelas indicam baixa sobreposição para as quais podem ser gerados resultados ruins nos dois projetos. As áreas verdes indicam uma sobreposição de mais de 5 imagens.



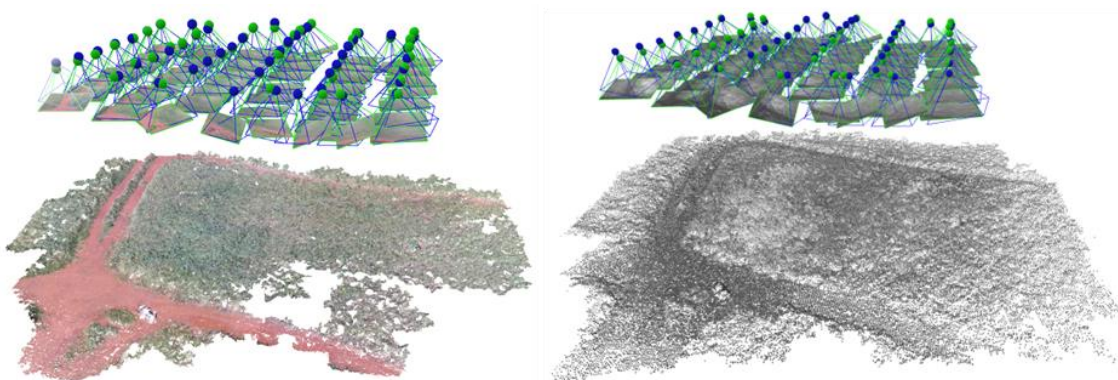
**Figura 4.42** – Número de imagens sobrepostas para cada pixel do mosaico RGB e multiespectral. Maior número de imagens sobrepostas melhora a precisão [PIX4DMAPPER].

Quando o número de pontos correspondentes encontrados na Etapa 1 não é suficiente, podemos executar esta etapa novamente, aumentando ou diminuindo a escala da imagem na qual os *keypoints* são extraídos, que, conseqüentemente, aumenta ou diminui o tempo de execução da Etapa 1. Na escala 1 (tamanho original da imagem) ou escala 2 (dobro do tamanho), o processamento é mais lento que nas escalas 1/2, 1/4 ou 1/8.

Utilizando a escala 1, obtivemos uma média de 79.372 *keypoints* por imagem, 3.549 pontos correspondentes por imagem calibrada, 197.250 pontos 3D densificados e 48 de 63 imagens calibradas (76%). As imagens não calibradas geometricamente na [Figura 4.43a](#) geraram falhas na nuvem de pontos. Para contornar este problema, testamos outras escalas, obtendo melhor resultado na escala 1/4, com uma média de 79.363 *keypoints* por imagem, 4.067 pontos correspondentes por imagem calibrada, 227.535 pontos 3D densificados e 55 de 63 imagens calibradas (82%) ([Figura 4.43b](#)). Antes do processamento definitivo da Etapa 1, removemos 8 imagens não calibradas e 7 imagens calibradas na subida do VANT devido à variação de altitude de voo, que causam variações na escala dos objetos. Com isso, obtivemos o projeto MosaicoRGB com 48 imagens calibradas das 63 imagens coletadas, assim como o projeto MapasReflectância com 192 imagens multiespectrais calibradas (4 bandas x 48) das 252 imagens coletadas (4 bandas x 63), listadas na [Tabela 4.2](#). A [Figura 4.44](#) mostra a nuvem de pontos 3D densificada dos dois projetos após a execução da densificação de pontos na Etapa 2.



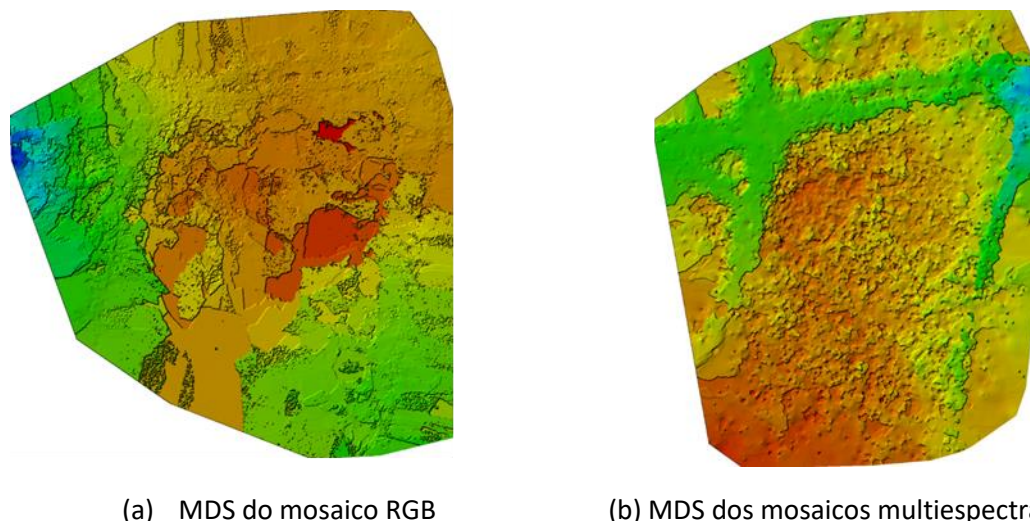
**Figura 4.43** – Problemas na etapa de calibração, gerando falhas no processamento da nuvem de pontos. Solução com diminuição da escala da imagem para cálculo de *keypoints*.



**Figura 4.44** – Nuvens de pontos 3D densificados dos projetos RGB e multiespectral.

#### 4.4.2 Modelo digital de superfície RGB e multiespectral

Na terceira etapa, os MDSs dos dois projetos foram gerados para ortorretificação do mosaico RGB e mapas de reflectância multiespectrais. A [Figura 4.45](#) mostra os MDSs em *raster* de baixa resolução para pré-visualização. Não salvamos o MDS de alta resolução que foi utilizado apenas internamente para ortorretificação das imagens.



**Figura 4.45** – Modelos digitais de superfície (MDS) de baixa resolução do conjunto de dados do campo experimental de cana-de-açúcar.

#### 4.4.3 Ortomosaico RGB e mapas de reflectância multiespectrais

Na Etapa 4 foram gerados um ortomosaico RGB e quatro mapas de reflectância multiespectrais, após ortorretificação das respectivas imagens RGB e multiespectrais sobre o MDS correspondente. A [Tabela 4.3](#) mostra as informações do ortomosaico RGB e mapas de reflectância multiespectrais. Os mapas NIR e *Red* foram usados para gerar um mapa de índice NDVI, descrito na [Seção 4.4.4](#).

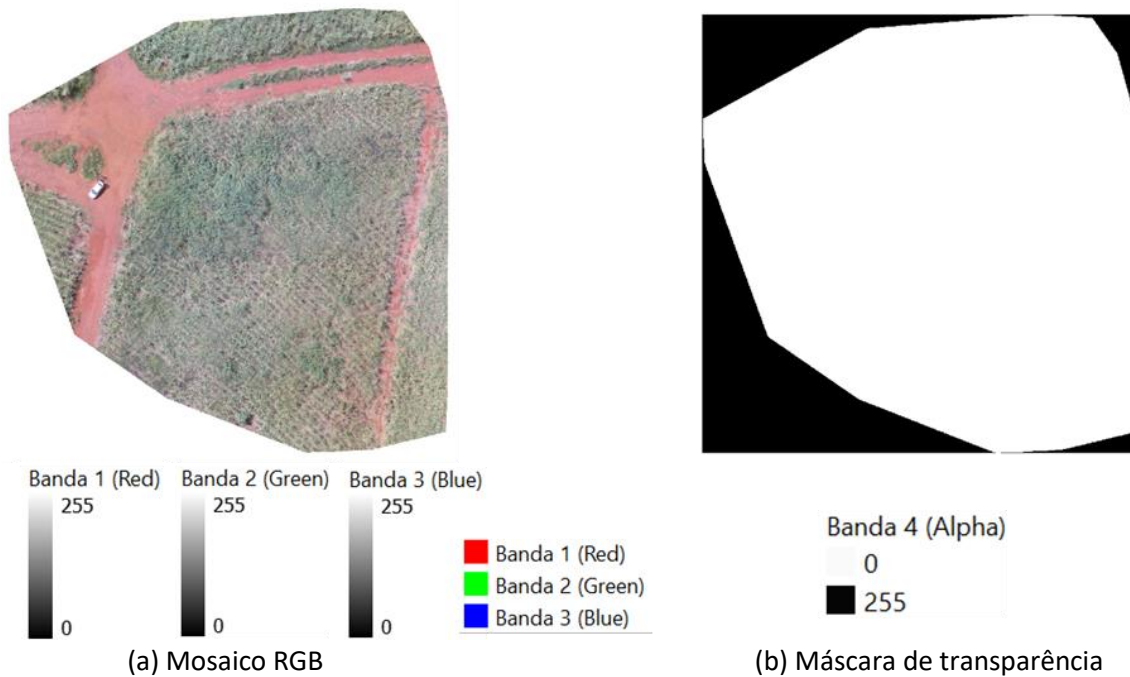
**Tabela 4.3** – Informações do ortomosaico RGB e mapas multiespectrais.

Arquivo GeoTIFF	Resolução espacial (pixels)	Banda + Máscara binária	Tipo de dados	GSD médio (cm/pixel)	Área de cobertura (km <sup>2</sup>   ha)
<b>Mosaico RGB</b>	8.196 × 8.260	3 + 1	Byte	1,16	0,007   0,7183 <sup>7</sup>
<b>Mapa Green</b>	2.470 × 2.903	1 + 1	Float32	4,63	0,013   1,2902
<b>Mapa Red</b>	2.470 × 2.903	1 + 1	Float32	4,63	0,013   1,2902
<b>Mapa Red-Edge</b>	2.470 × 2.903	1 + 1	Float32	4,63	0,013   1,2902
<b>Mapa NIR</b>	2.470 × 2.903	1 + 1	Float32	4,63	0,013   1,2902
<b>Mapa NDVI</b>	2.470 × 2.903	1 + 1	Float32	4,63	0,013   1,2902

<sup>7</sup> O mosaico RGB tem uma área de cobertura menor do que mapas de reflectância multiespectrais ([Figura 4.62](#)), devido a distorções das imagens aéreas RGB, que não foram ortorretificadas precisamente nas bordas do MDS, gerando descarte de dados nas bordas do mosaico RGB.

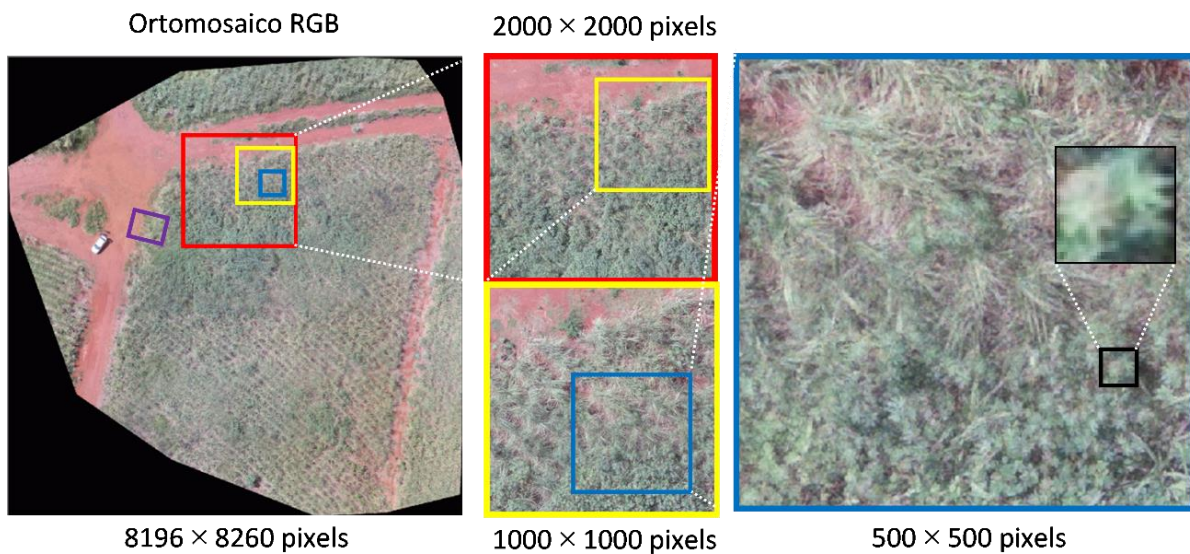


A [Figura 4.46a](#) mostra o mosaico RGB de  $8.196 \times 8.260$  pixels com um canal de transparência, onde pixels com valor 255 (preto) indicam a borda do mosaico. O mosaico tem GSD médio de 1,16 cm/pixel, cobrindo uma área de 0,007 km<sup>2</sup>.



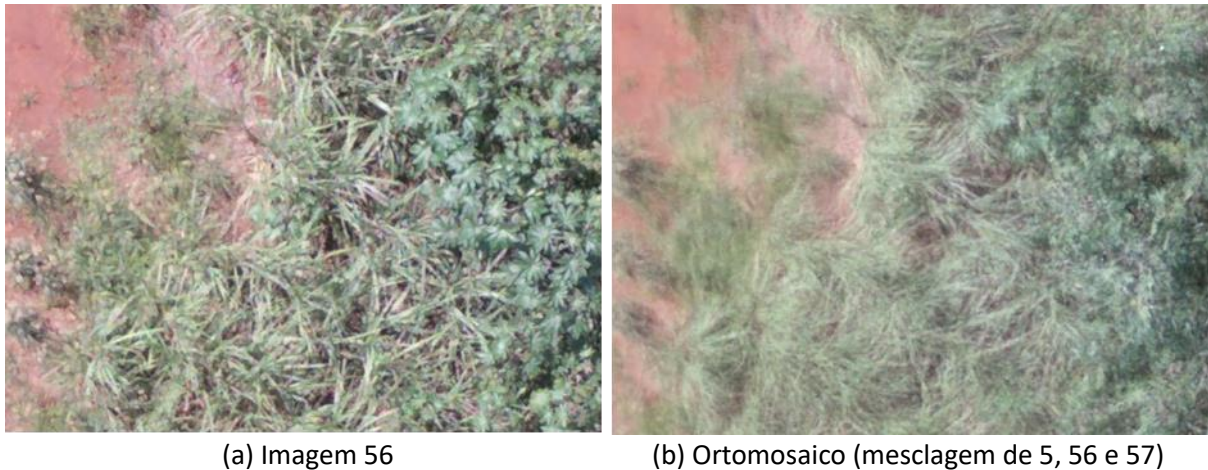
**Figura 4.46** – Ortomosaicos RGB de  $8.196 \times 8.260$  pixels, com GSD médio de 1,16 cm/pixel, gerado a partir de 48 imagens RGB de  $4.608 \times 3.456$  pixels.

A [Figura 4.47](#) mostra detalhes do mapa ortomosaico RGB. As caixas coloridas mostram uma visualização ampliada do ortomosaico em diferentes níveis de zoom. Na imagem à direita, as plantas daninhas de folha larga têm cerca de 30 pixels de largura (caixa preta), o que corresponde a 34,8 cm no solo com GSD de 1,16 cm/pixel.



**Figura 4.47** – Detalhes do ortomosaico RGB com GSD de 1,16 cm/pixel. A imagem esquerda mostra o mapa ortomosaico de  $8.196 \times 8.260$  pixels. As imagens dentro das caixas coloridas mostram uma ampliação de cada área em vários níveis de zoom. A caixa preta com duas folhas de mamona tem  $30 \times 30$  pixels, que corresponde a 34,8 cm.

Apesar da alta resolução espacial do ortomosaico RGB, a imagem pode apresentar perda de qualidade causada pelas distorções da imagem, conforme discutido na [Seção 4.3.4](#). Sendo assim, no ortomosaico pode ocorrer uma perda de detalhes capturadas pelas imagens aéreas originais devido ao processo de projeção ortogonal dos *pixels* das imagens sobre o MDS, bem como mesclagem da intensidade dos *pixels* das imagens sobrepostas. A [Figura 4.48b](#) mostra uma parte do mosaico, que foi gerada pela mesclagem de algumas imagens, degradando a qualidade visual de detalhes da vegetação.



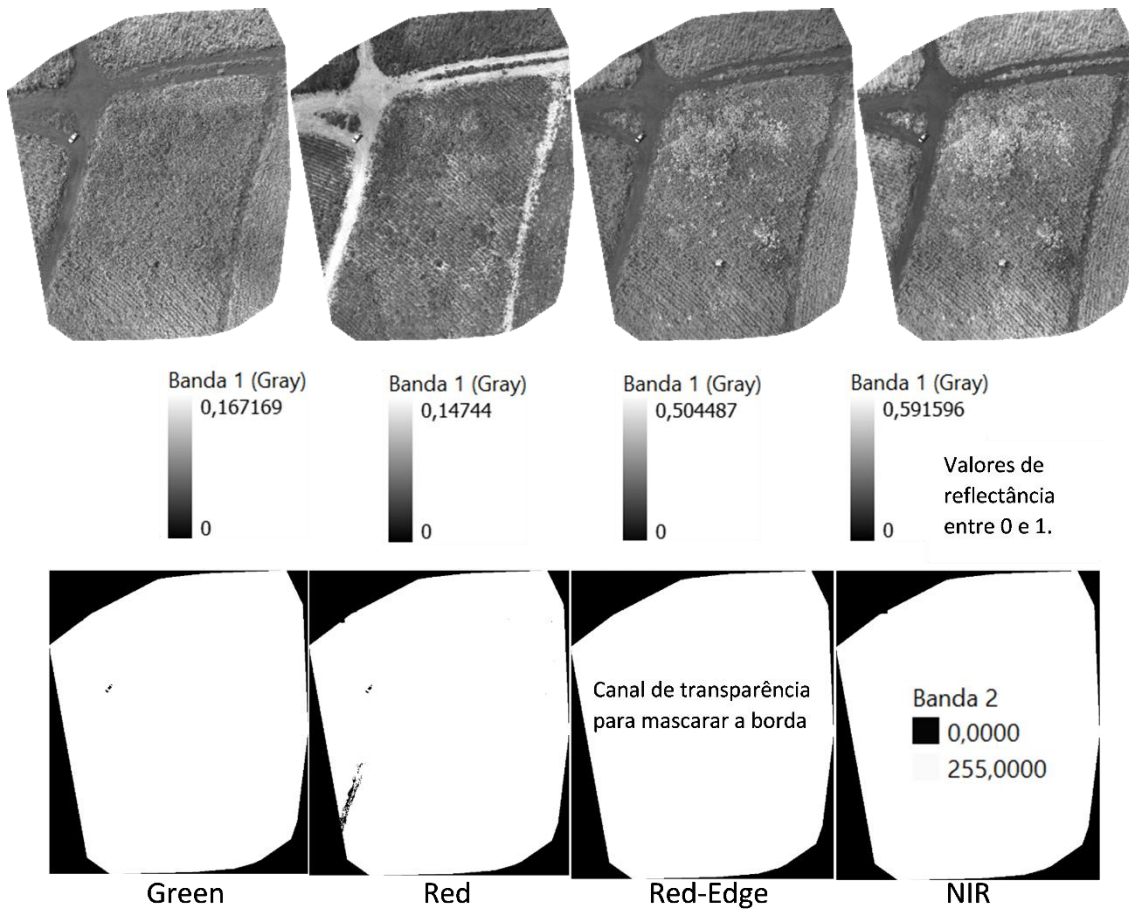
**Figura 4.48** – Detalhe da imagem original 56 (a); mesclagem dos *pixels* das imagens 5, 56 e 57 para gerar o mosaico RGB dentro do retângulo roxo da [Figura 4.47](#) (b).

Neste trabalho, como o mosaico é relativamente pequeno, utilizamos as imagens aéreas sobrepostas originais de alta resolução para aumentar a quantidade de amostras de dados de treinamento. No entanto, com objetivo de desenvolver redes neurais de segmentação semântica para aplicações reais de agricultura de precisão, utilizamos o mosaico para teste das redes. O principal motivo é que a utilização de ortomosaicos e mapas de reflectância georreferenciados apresenta algumas vantagens sobre as imagens originais. Por exemplo, é possível extrair medidas de uma cultura de forma quantitativa (em escala métrica) e mapear a localização precisa da distribuição de plantas daninhas em um campo para automatização das máquinas agrícolas. No entanto, o desempenho das redes de segmentação sobre o mosaico pode ser menor devido às distorções em relação às imagens aéreas originais de treinamento.

Na [Figura 4.49](#), cada mapa de reflectância *Green*, *Red*, *Red-Edge* e *NIR* ( $2.470 \times 2.903$  *pixels*) foi gerado a partir das 48 imagens multiespectrais ortonormalizadas correspondentes, com um GSD aproximado de  $4,63$  *cm/pixel*, cobrindo uma área de  $0,013$   $\text{km}^2$ . Os arquivos foram gerados em formato GeoTIFF com compressão LZW, com tipo de dados de ponto flutuante de 32 bits, com 2 canais. O primeiro canal armazena os valores de reflectância entre 0 e 1, e o segundo canal (de transparência) indica os valores válidos e inválidos do mapa gerado no PIX4Dmapper, servindo como uma máscara de borda.

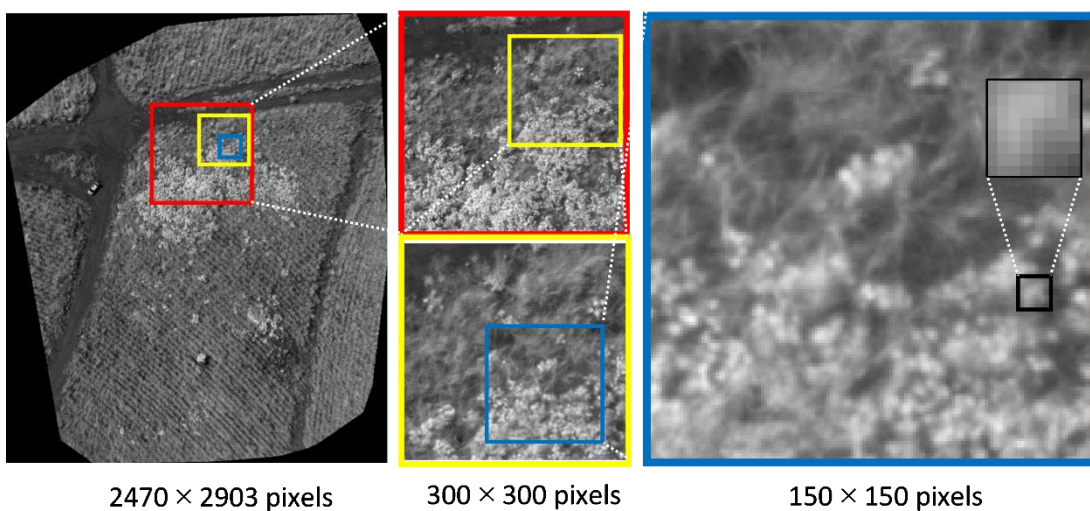
A [Figura 4.50](#) mostra detalhes do mapa de reflectância *Red-Edge* com GSD de  $4,63$  *cm/pixel*. As plantas daninhas aparecem destacadas na imagem multiespectral, o que facilita a discriminação. No entanto, a menor resolução espacial pode dificultar a visualização de características morfológicas individuais para identificação da espécie de planta ou discriminar plantas muito pequenas no início da fase de crescimento.





**Figura 4.49** – Mapas de reflectância e máscaras das quatro bandas espectrais (Green, Red, Red-Edge e NIR) com  $2.470 \times 2.903$  pixels cada, cobrindo uma área de  $0,013 \text{ km}^2$ , com GSD de  $4,63 \text{ cm/pixel}$ , gerado a partir de  $4 \times 48$  imagens multispectrais de  $1.280 \times 960$  pixels.

Mapa de Reflectância Red-Edge  $600 \times 600$  pixels

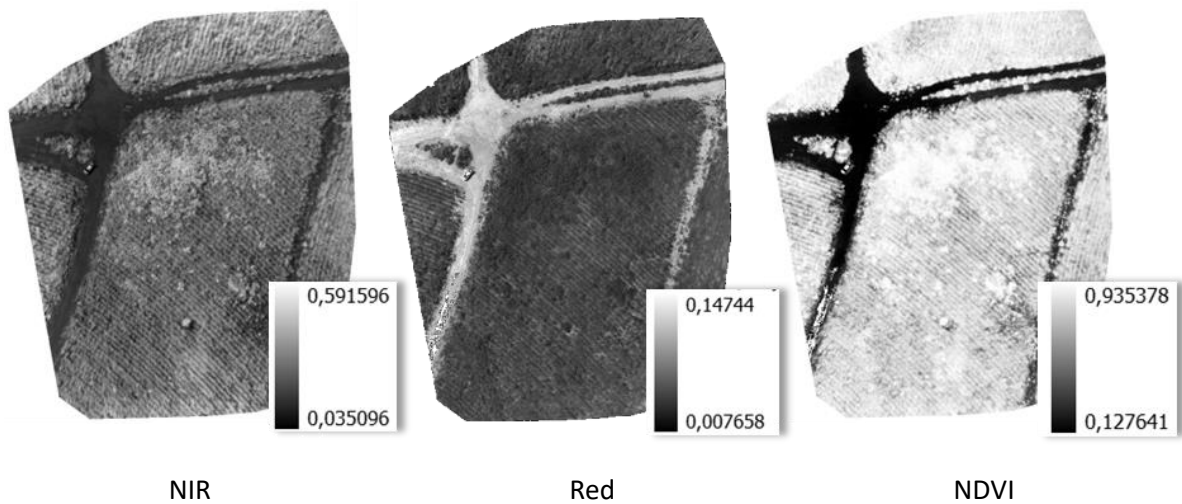


**Figura 4.50** – Detalhes do mapa de reflectância Red-Edge com GSD de  $4,63 \text{ cm/pixel}$ . A imagem esquerda mostra o mapa de  $2.470 \times 2.903$  pixels. As caixas coloridas ampliam cada área em vários níveis de zoom. A caixa preta com folha de mamona tem  $8 \times 8$  pixels, que corresponde a  $37 \text{ cm}$ .



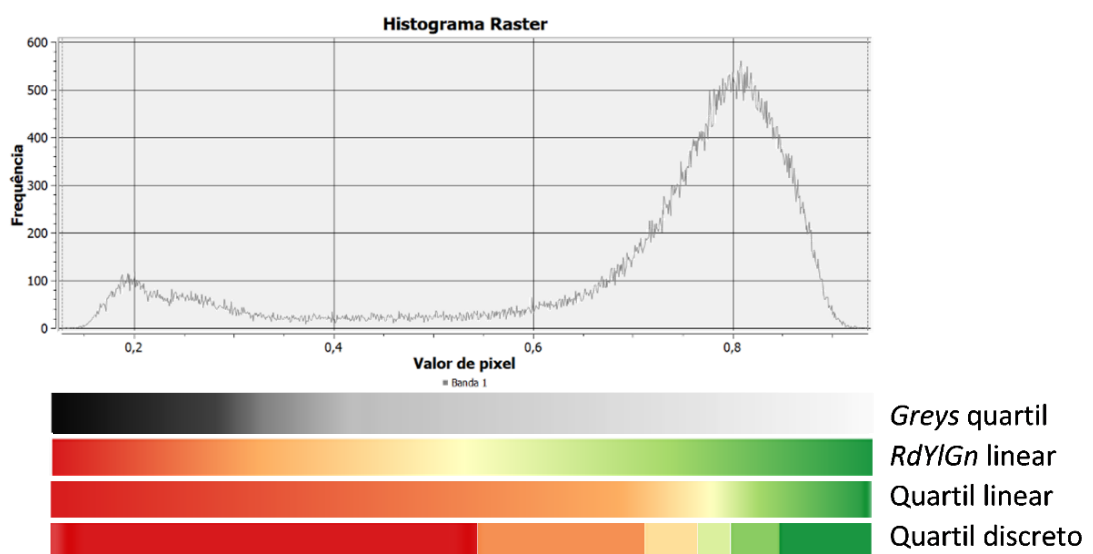
#### 4.4.4 Mapas de índice de vegetação

Na [Figura 4.51](#), mostramos dois mapas de reflectância – NIR e Red – utilizados para cálculo do mapa de índice NDVI no PIX4Dmapper. Onde há alta reflectância dentro da faixa NIR e baixa reflectância da faixa espectral Red pela absorção de clorofila, temos um alto valor de índice no mapa NDVI, destacando a vegetação.



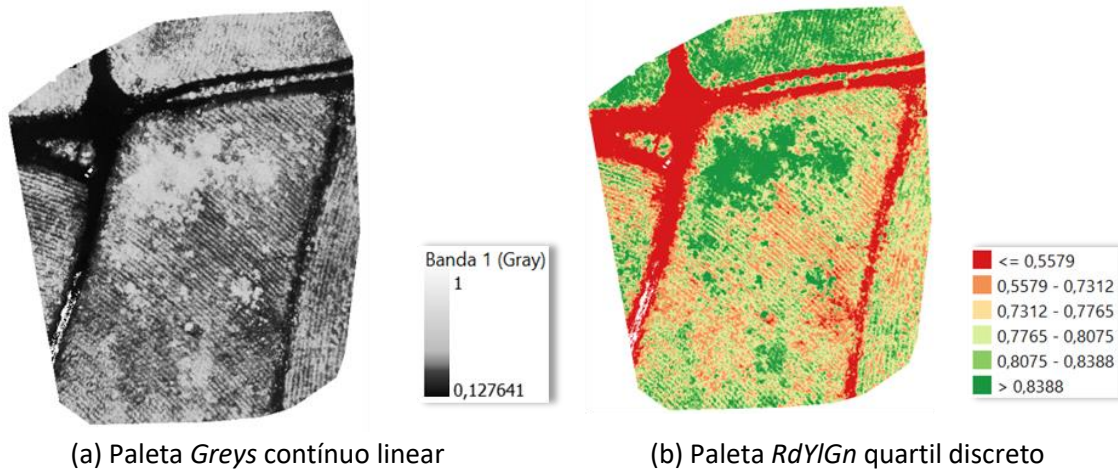
**Figura 4.51** – Mapa de índice NDVI visualizado em tons de cinza, calculado com base nos mapas de reflectância NIR e Red.

Como o PIX4Dmapper requer uma licença para cálculo e visualização do mapa NDVI, usamos o QGIS 3.28 Firenze [[QGIS](#)], um software livre de sistema de informação geográfica, que permite a visualização, edição e análise de dados georreferenciados. A [Figura 4.52](#) mostra o histograma do mapa NDVI e as paletas de cores utilizadas para visualização do mapa em tons de cinza ou pseudocor (falsa-cor).



**Figura 4.52** – Histograma do mapa de índice NDVI e paletas de cores para visualização em falsa-cor.

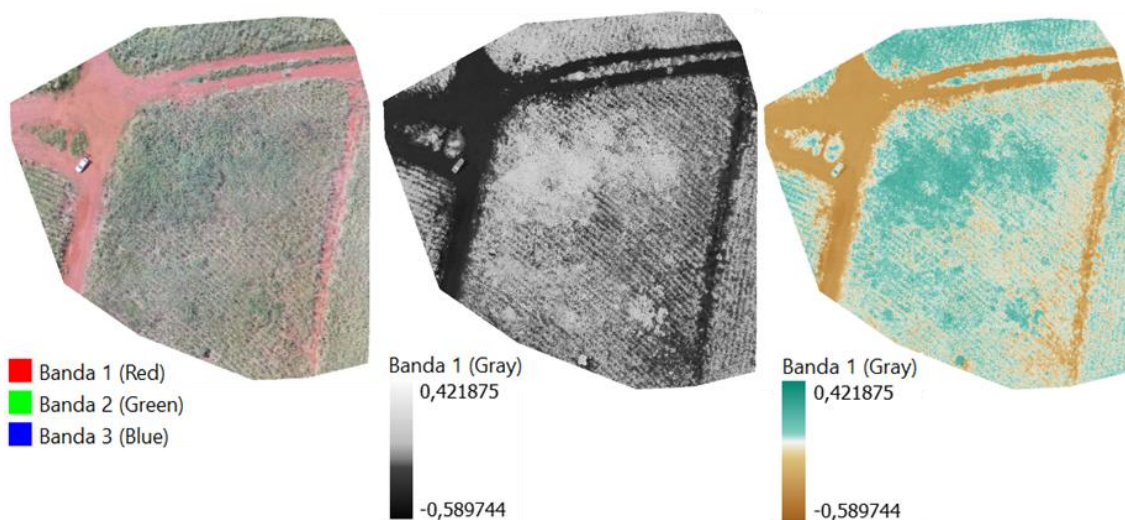
Selecionando a paleta de cores *Greys* no modo quartil com interpolação linear (Figura 4.53a) ou *RdYlGn* modo quartil com método discreto (Figura 4.53b), podemos aumentar o contraste e destacar as plantas daninhas. Por exemplo, na Figura 4.53b, as áreas com índice NDVI com valor 0 a 0,55 são exibidos em vermelho, 0,55 a 0,77 em laranja, 0,77 a 0,83 em verde-claro e acima de 0,83 em verde-escuro.

(a) Paleta *Greys* contínuo linear(b) Paleta *RdYlGn* quartil discreto

**Figura 4.53** – Mapa de índice NDVI calculado e visualizado em falsa-cor no QGIS.

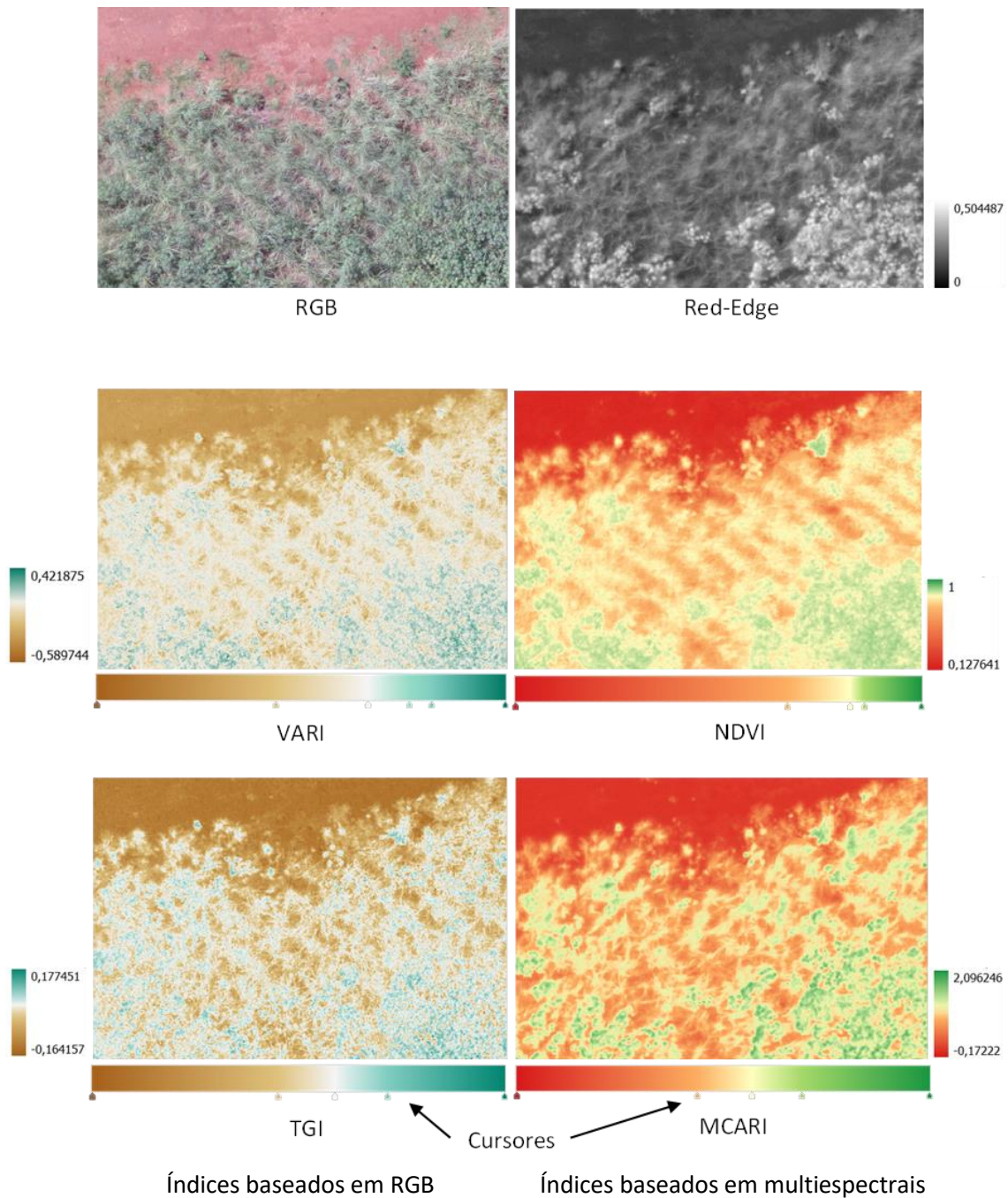
Diferentemente do índice NDVI, que é baseado em sensores multiespectrais, outros índices podem ser calculados baseados nas bandas espectrais visíveis (R, G e B) do sensor RGB. Por exemplo, o índice VARI =  $(G - R) / (G + R - B)$  é destinado para detecção de áreas de estresse nas lavouras e analisa o nível de 'verde' capturado nas imagens. No entanto, mapas de índice no espectro visível não buscam a substituição do NDVI.

Como podemos notar na Figura 4.54, o mapa VARI visualizado com uma paleta de cores linear com alto contraste permite uma melhor visualização da distribuição das plantas daninhas na cultura de cana-de-açúcar (em verde-escuro), quando comparado ao mosaico RGB, onde a cultura e daninha têm o mesmo espectro de onda visível. No entanto, quando comparado ao *groundtruth* na Figura 3.1, observa-se que este índice identifica plantas daninhas em verde-escuro no topo do mosaico, onde na verdade não existem.



**Figura 4.54** – Ortomosaico RGB e mapa de índice VARI, calculado com base nas bandas espectrais visíveis RGB, visualizado em tons de cinza ou paleta de cores.

A [Figura 4.55](#) mostra um recorte do mosaico RGB e do mapa de reflectância *Red-Edge* que destaca as plantas daninhas (mamona), assim como alguns exemplos de mapas de índices de vegetação gerados a partir das bandas RGB e multiespectrais, visualizados em alto contraste com diferentes especificações das paletas de cores, pelo deslizamento dos cursores sobre a barra na parte inferior da imagem. No entanto, a especificação das paletas para discriminação da vegetação e daninhas é uma tarefa manual, tornando o processo dispendioso e de difícil automatização, principalmente usando os mapas de índices de baixo contraste calculados a partir das bandas RGB do espectro visível.



**Figura 4.55** – Recorte do ortomosaico RGB e mapa *Red-Edge*, com mapas de índices visualizados em alto contraste com diferentes especificações das paletas de cores.



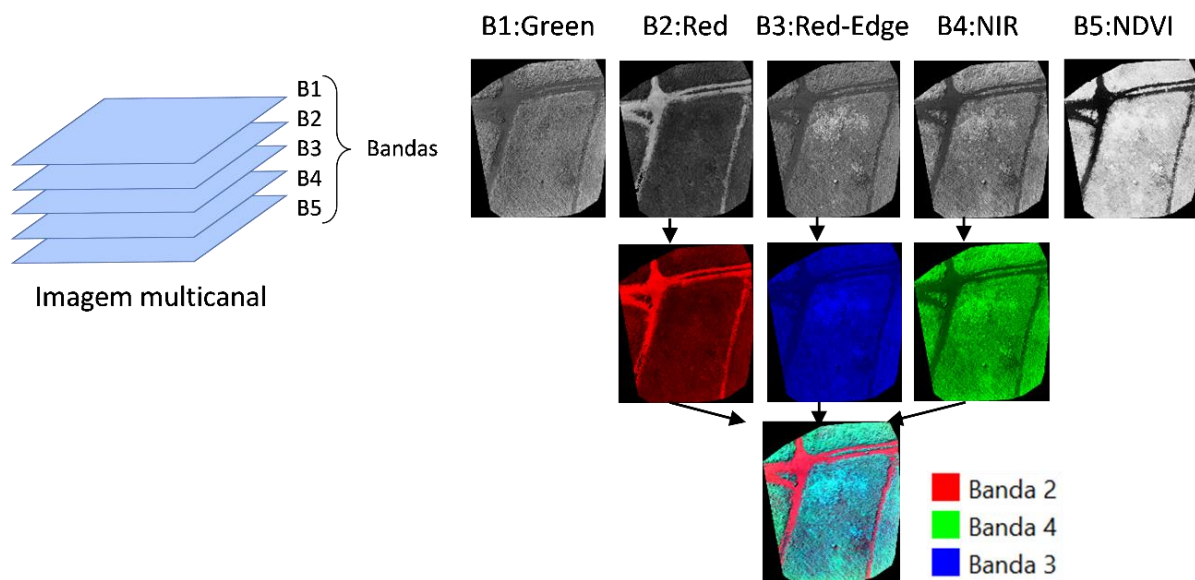
### 4.4.5 Imagem multicanal e composição colorida

Como todos os mapas de reflectância multiespectrais e os mapas de índice georreferenciados gerados pelo PIX4Dmapper têm mesma resolução espacial e foram alinhados com precisão, é possível gerar mapas multicanais, ou seja, compostos por vários canais com uma combinação de bandas dos mapas multiespectrais e mapas de índices de vegetação. Utilizamos o programa QGIS para empilhar as bandas nos canais da imagem, gerando um único arquivo GeoTIFF multibanda, como mostra a [Tabela 4.4](#). Este mapa multicanal pode ser usado como entrada das redes de segmentação semântica densa.

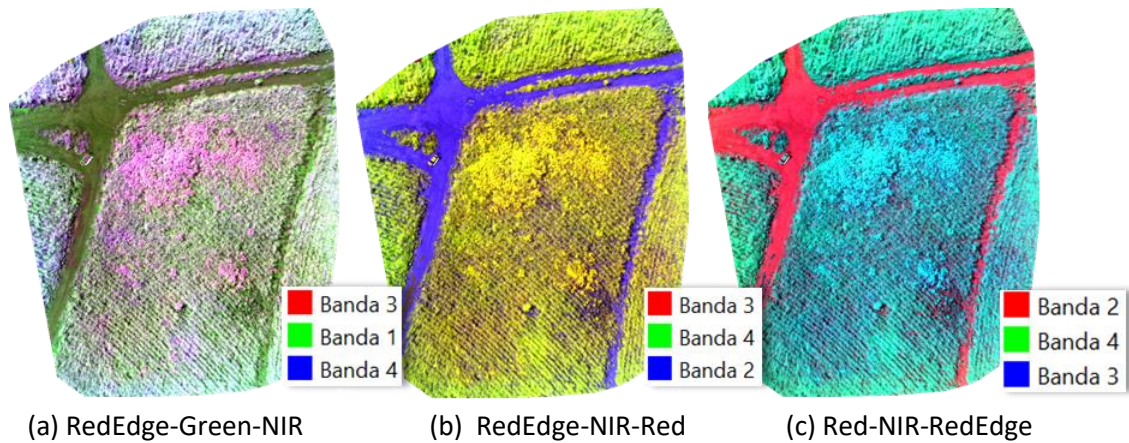
**Tabela 4.4** – Mapa multicanal com cinco bandas espectrais e índice de vegetação NDVI.

Banda	Espectro	Resolução espacial	Tipo de dado	Mín	Máx
<b>B1</b>	Green	2.470 × 2.903	Float32	0.0000000000	0.1671687216
<b>B2</b>	Red	2.470 × 2.903	Float32	0.0000000000	0.1474399567
<b>B3</b>	Red-Edge	2.470 × 2.903	Float32	0.0000000000	0.5044869184
<b>B4</b>	NIR	2.470 × 2.903	Float32	0.0000000000	0.5915962458
<b>B5</b>	NDVI	2.470 × 2.903	Float32	0.1276406348	0.9353784919

Para interpretação do mapa multicanal por cores, geramos algumas composições coloridas pela combinação de diferentes bandas espectrais nos canais RGB. A [Figura 4.56](#) mostra a composição das bandas 2, 4, 3 (*Red, NIR, Red-Edge*) do mapa multicanal da [Tabela 4.4](#). Várias composições coloridas podem ser facilmente geradas e visualizadas no QGIS, o que possibilita uma maior diferenciação entre solo exposto, cana-de-açúcar e plantas daninhas, como mostrado na [Figura 4.57](#).



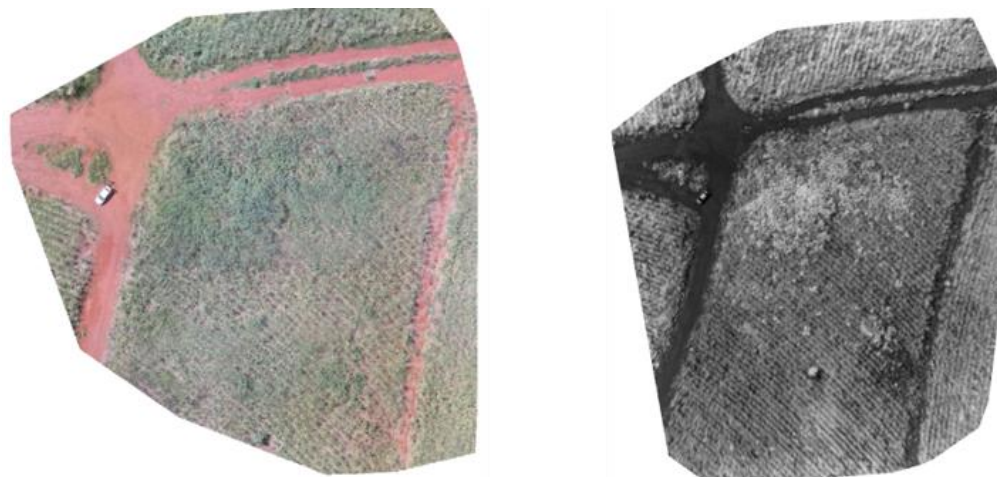
**Figura 4.56** – Composição colorida das bandas espectrais 2, 4, 3 (*Red, NIR, Red-Edge*).



**Figura 4.57** – Exemplos de composições coloridas multiespectrais nos canais RGB.

Conforme discutido na [Seção 3.1](#) e empregado nos trabalhos correlatos das [Seções 3.3](#) e [3.4](#), mapas multicanais com diferentes bandas espectrais permitem diferentes estratégias de entrada em redes de classificação ou segmentação semântica, como mostrado na [Tabela 3.2](#). Portanto, usando uma câmera multiespectral, composta por um sensor RGB visível e 4 sensores multiespectrais, seria possível obter combinações multicanais com 7 canais ou mais. Porém, isto só é viável quando utilizadas câmeras multiespectrais que geram ortomosaicos e mapas de reflectância geometricamente alinhados. Por exemplo, a câmera XM5 da empresa Xrobots<sup>8</sup> captura todas as bandas espectrais visíveis e infravermelhas (R, G, B, Red-Edge e NIR) em um único sensor com a mesma configuração de obturador (*global shutter*) e resolução espacial de 3.000 x 2.000 *pixels*, armazenando uma banda espectral por imagem em formato GeoTIFF de 16 bits. Desta forma, é possível gerar mosaicos RGB e multiespectrais, combinações multicanais e uma grande variedade de índices vegetativos [[HOLLER et. al. 2022](#)].

No entanto, especificamente no caso da câmera Sequoia, o ortomosaico (gerado a partir do sensor RGB) e os mapas de reflectância (gerados a partir dos sensores multiespectrais) não são alinhados geometricamente, como mostra a [Figura 4.58](#). Os motivos deste desalinhamento serão explicados na [Seção 4.4.5.3](#), o que dificulta a criação de mapas multicanais do tipo RGB + Multiespectral. A seguir, apresentamos mais informações sobre os tipos de desalinhamento das imagens originais e dos mosaicos da câmera Sequoia, que dificultam o empilhamento multicanal.



**Figura 4.58** – Mosaicos RGB e mapa de reflectância NIR desalinhados.

<sup>8</sup> Especificação da câmera XM5: <https://xrobots.com.br/cameras/xm5/>.

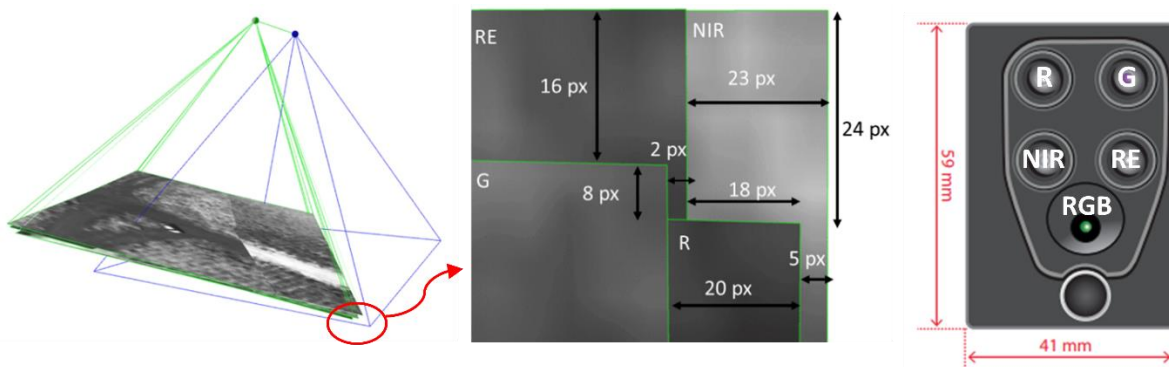


#### 4.4.5.1 Desalinhamento entre imagens originais multiespectrais

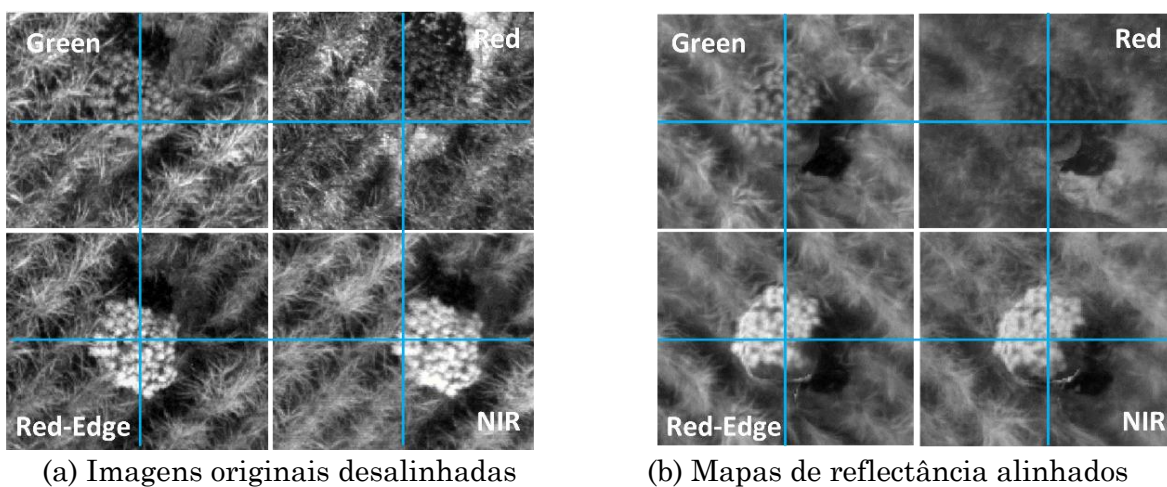
As imagens aéreas originais multiespectrais da câmera Sequoia requerem um pré-processamento para alinhamento das bandas espectrais, antes de serem empilhadas em uma única imagem multicanal, a fim de serem usadas como entrada em redes neurais convolucionais. Este desalinhamento ocorre devido à posição das lentes óticas na câmera multiespectral Sequoia, como pode ser examinado visualmente na [Figura 4.59](#).

Um bom alinhamento de todas as bandas é crucial para análise subsequente da imagem multicanal. Os objetos de vegetação presentes no campo no estágio inicial são pequenos e, como consequência, o desalinhamento inclui *pixels* que não pertencem aos objetos de interesse, reduzindo o sucesso de redes de segmentação semânticas treinadas a partir das imagens originais multiespectrais. Nas [Figuras 4.59 e 4.60a](#), podemos ver os deslocamentos entre bandas. Como as imagens multiespectrais têm GSD médio de 4,63 cm/*pixel*, o maior deslocamento de 24 *pixels* equivale a um deslocamento de 111 cm no solo.

Entretanto, nos mapas de reflectância multiespectrais, as bandas são alinhadas corretamente pelo software PIX4Dmapper [[PIX4DMAPPER](#)], como mostra a [Figura 4.60b](#). Isso permite criar mapas de índices de vegetação e mapas multicanais multiespectrais, que podem ser alimentados diretamente na entrada de redes de segmentação semântica de imagens.



**Figura 4.59** – Medidas dos deslocamentos entre as bandas Green, Red, Red-Edge e NIR devido à posição das lentes na câmera multiespectral Sequoia.

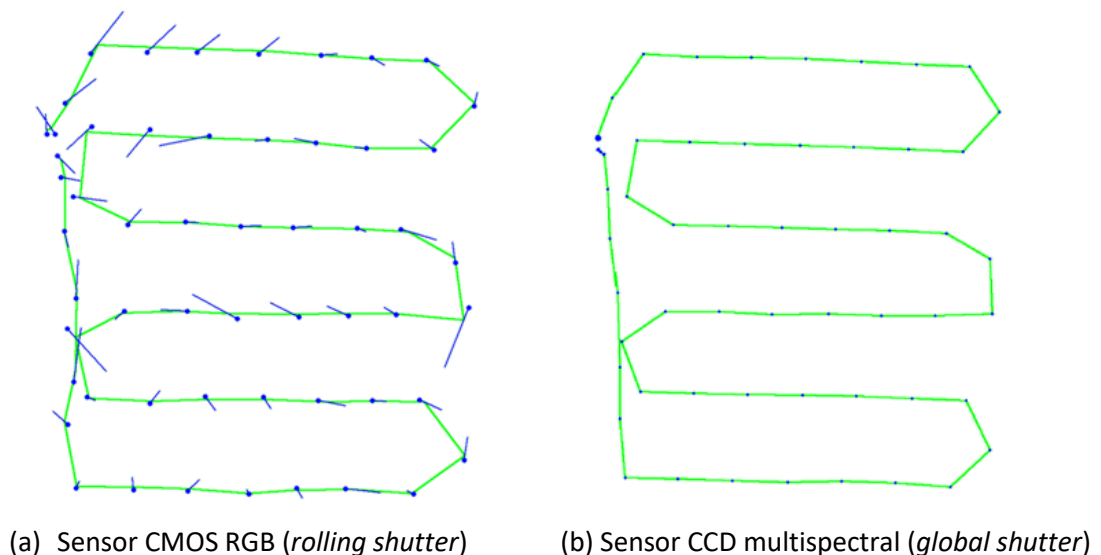


**Figura 4.60** – Desalinhamento espacial entre as bandas multiespectrais Green, Red, Red-Edge e NIR nas imagens originais (a); Mapas de reflectância multiespectrais alinhados.

#### 4.4.5.2 Desalinhamento entre imagens originais RGB e multiespectrais

Além da câmera Sequoia usar lentes diferentes (lente perspectiva no sensor RGB e lente olho-de-peixe nos sensores multiespectrais) com parâmetros de distorções radiais distintos e resoluções espaciais diferentes, o sensor RGB usa um obturador rolante (*rolling shutter*), enquanto os sensores multiespectrais usam um obturador global (*global shutter*).

O PIX4Dmapper corrige distorções das lentes e modela o obturador rolante para fazer correções nas imagens, de acordo com o deslocamento do VANT, como mostra a [Figura 4.61](#). No entanto, mesmo após a correção destas distorções na Etapa 1, as imagens corrigidas RGB não podem ser empilhadas com as imagens corrigidas multiespectrais para gerar imagens multicanais. Além disso, o mapa ortomosaico e os mapas de reflectância gerados a partir das imagens corrigidas continuam não sendo perfeitamente alinhados devido a outros fatores descritos a seguir.



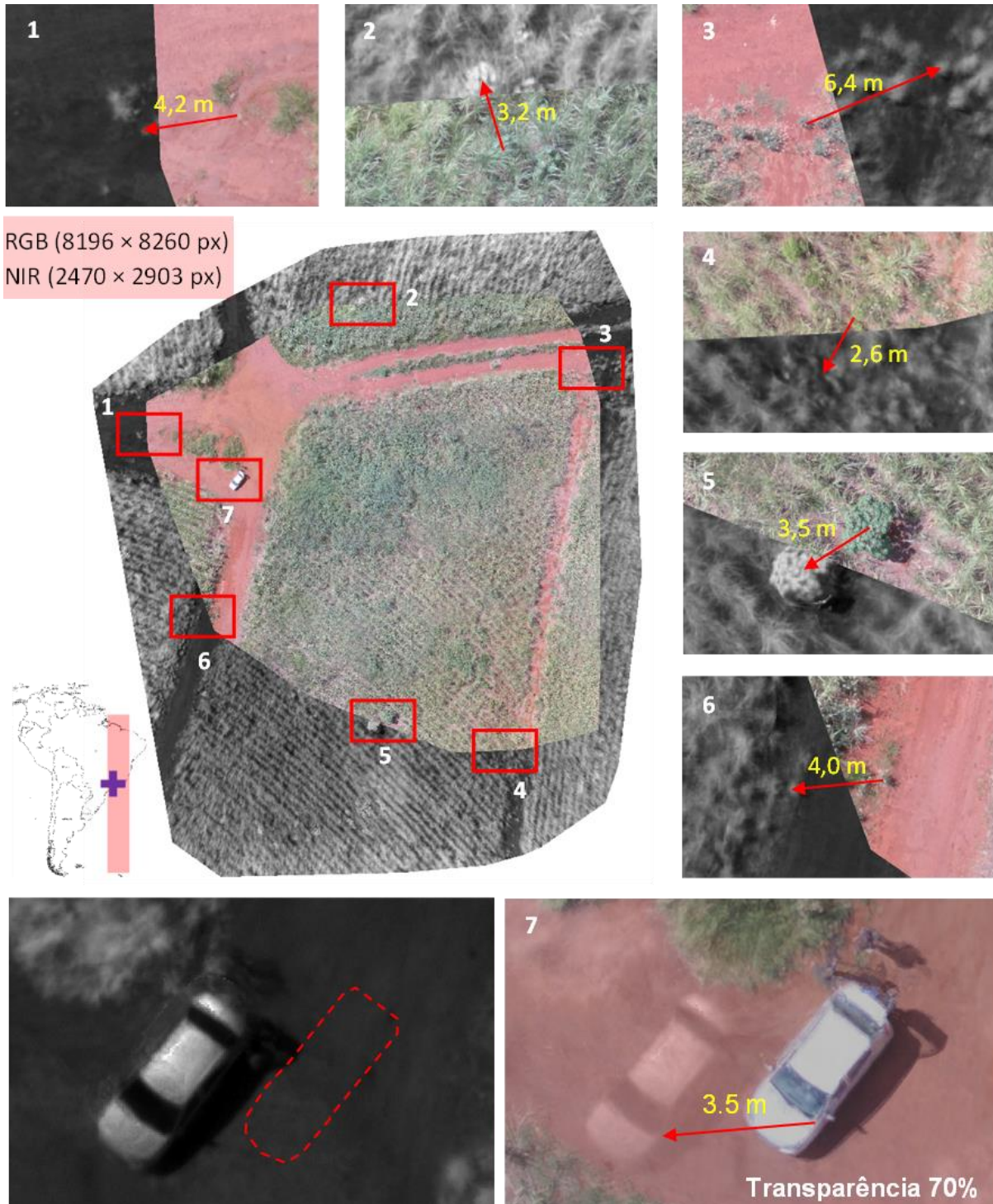
**Figura 4.61** – (a) *Estimativa de movimento de câmera com rolling shutter. A linha verde mostra o plano de voo e as posições das imagens. Os pontos azuis representam a posição da câmera no início da exposição. As linhas azuis representam o movimento da câmera durante a leitura do obturador rolante;* (b) *Imagens RGB capturadas com sensor do tipo global shutter.*

#### 4.4.5.3 Desalinhamento entre mosaico RGB e mapas de reflectância

O PIX4Dmapper realiza o processamento das imagens RGB e multiespectrais da câmera Sequoia em dois projetos distintos, gerando um mapa ortomosaico RGB e mapas de reflectância georreferenciados com diferentes características e tamanhos.

Como o ortomosaico RGB e os mapas de reflectância não foram alinhados geograficamente com precisão, uma vez que dois pontos correspondentes em diferentes posições do ortomosaico e do mapa de reflectância apresentam-se deslocados por alguns metros, como mostrado na [Figura 4.62](#), não podemos empilhar diretamente o mosaico RGB com os mapas multiespectrais da câmera Sequoia.

A **Figura 4.62** mostra uma visualização do ortomosaico RGB georreferenciado ( $8.196 \times 8.260 \text{ pixels}$ ) sobreposto ao mapa NIR ( $2.470 \times 2.903 \text{ pixels}$ ) no QGIS, com alguns deslocamentos medidos em metros em diferentes localizações do mosaico<sup>9</sup>. Sendo assim, somente câmeras multiespectrais com alinhamento de todas as bandas permitem fazer uma composição do mosaico RGB com os mapas de reflectância multiespectrais.



**Figura 4.62** – Deslocamentos entre ortomosaico RGB e mapa de reflectância NIR.

<sup>9</sup> O sistema de referência de coordenadas (SRC) utilizado foi EPSG:32723 - WGS 84/UTM zone 23S (*World Geodetic System 1984*), com unidade em metros, pelo método *Universal Transverse Mercator* (UTM), com precisão limitada de no máximo 2 metros.





# Capítulo 5

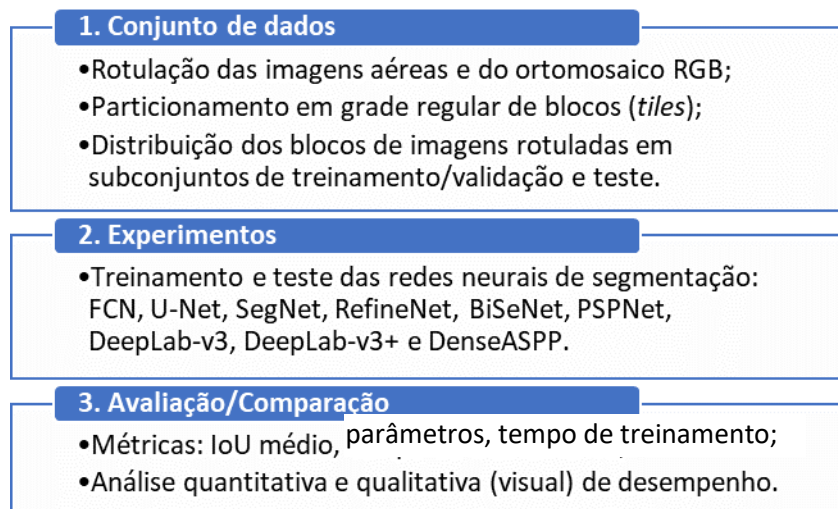
## Segmentação semântica de imagens aéreas

Este capítulo apresenta uma avaliação de algumas redes neurais profundas – descritas no [Capítulo 2](#) – treinadas em nossa tarefa específica de segmentação semântica de plantas daninhas em uma cultura de cana-de-açúcar. As redes foram treinadas com imagens aéreas RGB de alta resolução de VANT e testadas em um ortomosaico gerado de acordo com informações do [Capítulo 4](#). O fluxo de trabalho para treinamento e avaliação das redes é dividido em três etapas, como resumido na [Figura 5.1](#).

Na primeira etapa, preparamos os conjuntos de dados, rotulando manualmente algumas imagens aéreas originais e o ortomosaico RGB em quatro classes: solo, cultura, daninha e gramínea. As imagens aéreas, o ortomosaico RGB, como também seus respectivos *groundtruths* são divididos em uma grade regular de pequenos blocos (*tiles*) de imagens. Por fim, os blocos das imagens aéreas foram distribuídos em subconjuntos de treinamento/validação e teste, enquanto os blocos do ortomosaico são usados apenas para teste das redes, conforme ilustrado na [Figura 5.2](#).

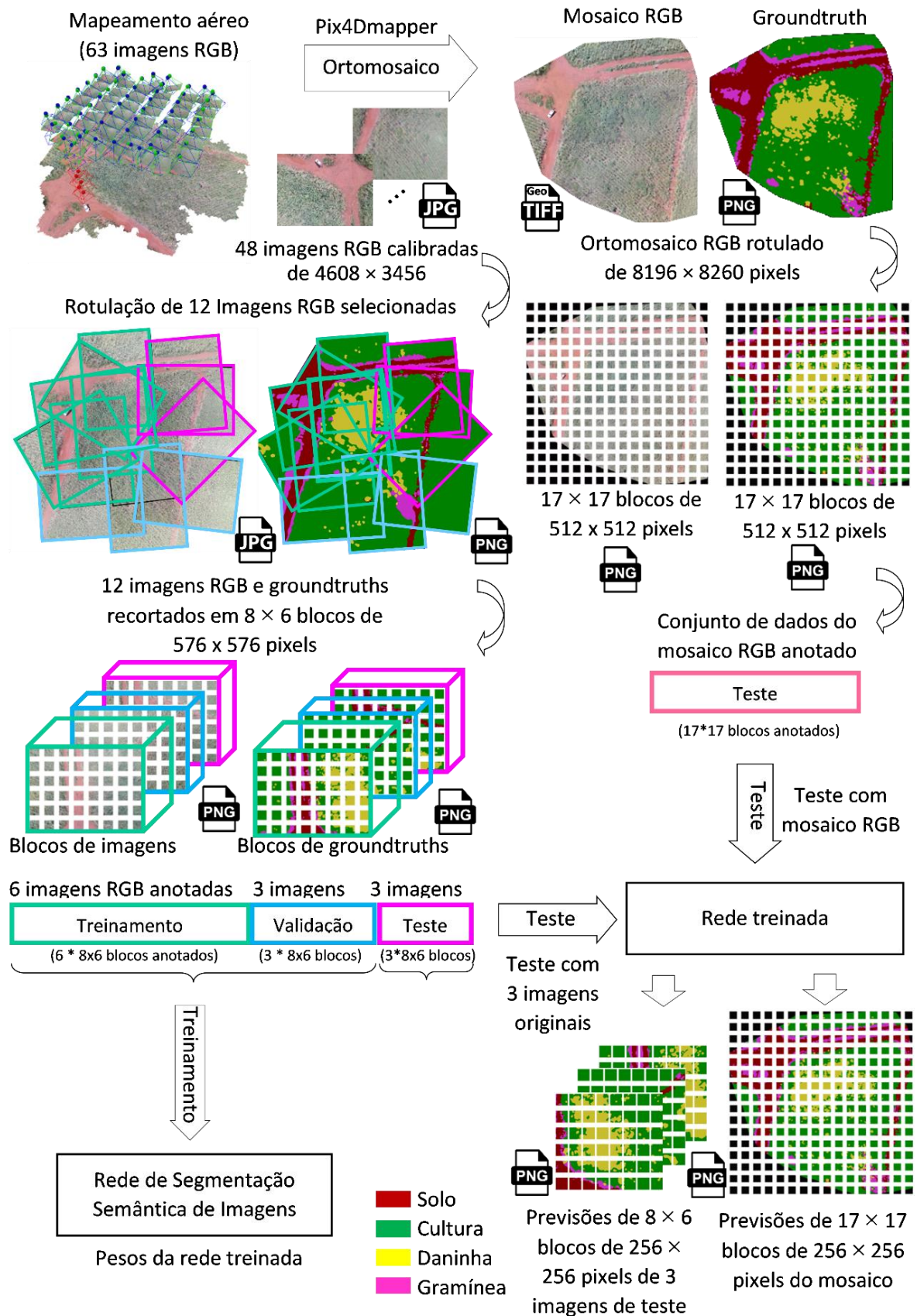
Na segunda etapa, selecionamos nove redes neurais para realizar experimentos na tarefa de segmentação semântica de plantas daninhas. Para isso, as redes foram treinadas e testadas em nossos conjuntos de dados.

Na última etapa, realizamos uma análise quantitativa das métricas de desempenho e uma análise qualitativa (visual) das imagens previstas, comparando os desempenhos das redes neurais nesta tarefa específica na área agrícola. A seguir, descrevemos os detalhes de cada etapa do fluxo de trabalho.



**Figura 5.1** – Etapas do fluxo de trabalho para avaliação das redes de segmentação.





**Figura 5.2** – Esquema geral para treinamento das redes de segmentação a partir das imagens aéreas originais e teste das redes a partir do mosaico RGB. Os groundtruths e previsões mostram os rótulos das classes solo, cultura, daninha e gramínea em cores.

## 5.1 Preparação do conjunto de dados

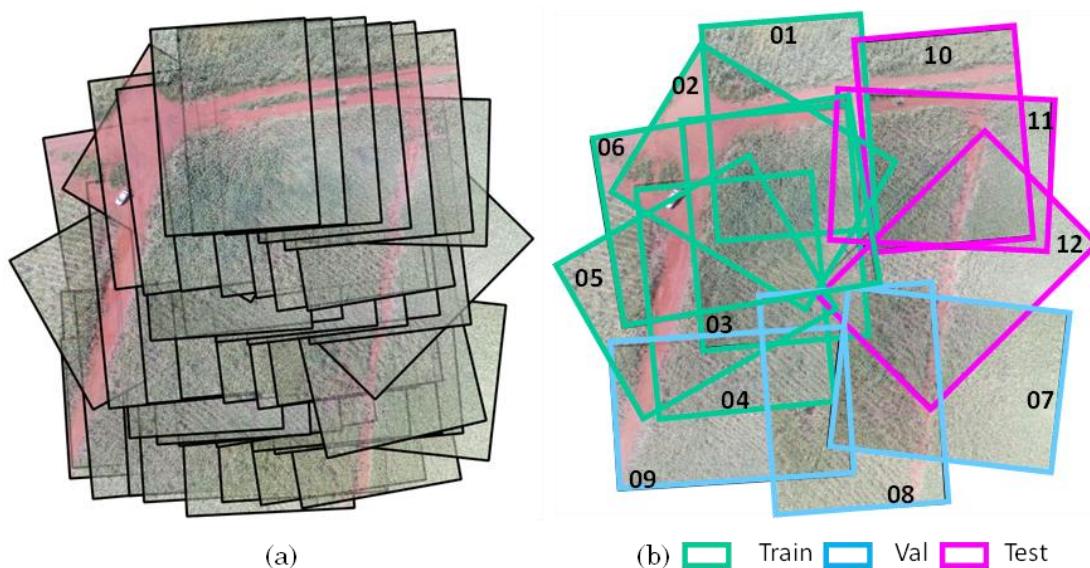
Geralmente, em aplicações de agricultura de precisão, o mosaico de alta resolução é dividido em partes para gerar os conjuntos de treinamento, validação e teste das redes de segmentação. No entanto, por causa de nosso conjunto pequeno e limitado de dados do ortomosaico RGB, utilizamos uma estratégia para aumentar naturalmente a quantidade de dados, usando as imagens aéreas originais (capturadas com sobreposição) para treinamento das redes de segmentação semântica, deixando o mosaico apenas para teste.

Apesar dos avanços recentes das CNNs, as imagens aéreas e os ortomosaicos de alta resolução são muito grandes para serem inseridos diretamente como entrada em uma rede convolucional padrão devido à limitação de memória da GPU. Por outro lado, a diminuição da resolução espacial com subamostragem na entrada da rede pode causar perda de propriedades importantes que permitem distinguir os tipos de vegetação. Este problema foi abordado por Sa *et al.* na rede WeedMap [SA *et al.* 2018], utilizando uma técnica de *tiling* (isto é, um particionamento do mosaico em uma grade de blocos retangulares) para gerar as subimagens sem subamostragem na entrada da rede. Um pós-processamento é usado para reconstrução da grade completa de blocos preditos na saída da rede.

A Figura 5.2 ilustra, de forma geral, os procedimentos que usamos para gerar os conjuntos de dados de treinamento, validação e teste a partir das imagens originais RGB, além de um conjunto de teste a partir do ortomosaico RGB. A seguir, detalhamos o processo de rotulação das imagens aéreas e do mosaico, bem como o particionamento das imagens aéreas e do mosaico utilizando a técnica de *tiling*.

### 5.1.1 Rotulação das imagens originais

Dentre o conjunto de 48 imagens originais RGB orientadas no espaço (Figura 5.3a), mas sem correção de distorções, selecionamos 12 imagens aéreas originais de  $4.608 \times 3.456$  pixels. A Figura 5.3b mostra: seis imagens selecionadas para treinamento (verde), três para validação (azul) e três para teste (lilás), correspondente a três áreas parcialmente disjuntas no terreno.

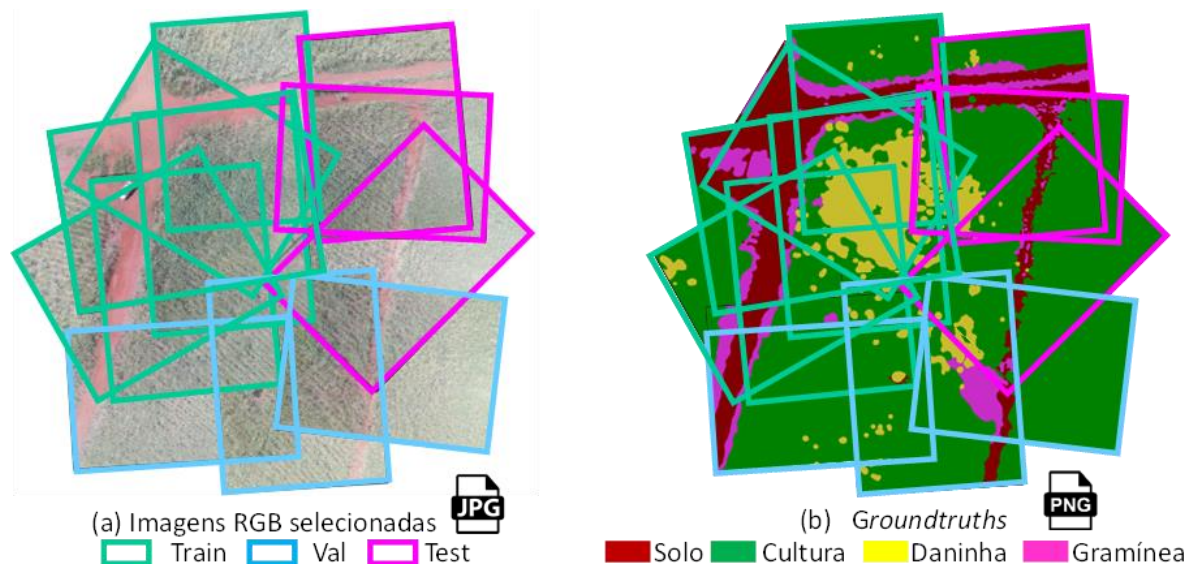


**Figura 5.3** – 48 imagens RGB calibradas (a) e 12 imagens selecionadas para treinamento (verde), validação (azul) e teste (lilás) das redes de segmentação.

Observa-se na [Figura 5.3](#) que, por causa da sobreposição das imagens, há redundância das amostras de dados de cada conjunto, uma vez que as mesmas áreas foram observadas em diferentes imagens ao longo do voo. No entanto, devido às variações de posição, orientação, ângulo de visão, condição de iluminação e altura de voo, estas amostras proporcionam um aumento natural de dados, de forma complementar ao aumento artificial de dados, para evitar o sobreajuste da rede neural.

Na tarefa de classificação semântica em nível de *pixel*, conjuntos de dados de treinamento de alta qualidade são essenciais. No entanto, muitas vezes é um desafio anotar imagens manualmente, sem orientação de um especialista, considerando que as ervas daninhas não são facilmente distinguíveis da cana-de-açúcar no espectro visível, nem mesmo nas imagens RGB de alta resolução. Para tentar minimizar este problema, rotulamos algumas imagens aéreas RGB, manualmente, em programas de manipulação de imagem (como Gimp ou Photoshop), com auxílio das imagens multiespectrais NIR e *Red-Edge*, que destacam, nitidamente, as plantas daninhas em nosso conjunto de dados.

Na [Figura 5.4](#), cada imagem RGB (em formato JPEG de 24 bits) foi anotada em um conjunto de rótulos semânticos pertencentes a quatro classes: solo, cultura de cana-de-açúcar, daninha de folhas largas e gramínea. Os *groundtruths* foram salvos em formato PNG (*Portable Network Graphic*) de 8 bits, com valores de *pixel* correspondentes a [0, 1, 2, 3] para cada classe. Os *groundtruths* foram salvos também em PNG colorido de 24 bits para facilitar a visualização, de acordo com uma tabela de cores RGB, sendo a classe solo representada em vermelho (130, 0, 0), cultura em verde (0, 130, 0), daninha em amarelo (200, 190, 45) e gramínea em rosa (200, 45, 200). Desta forma, é possível visualizar facilmente possíveis falhas na plantação e localizar espécies invasoras.



**Figura 5.4** – Rotulação manual de 12 imagens aéreas RGB sobrepostas de  $4.608 \times 3.456$  pixels, com seus *groundtruths* contendo quatro classes: solo, cultura, daninha e gramínea.

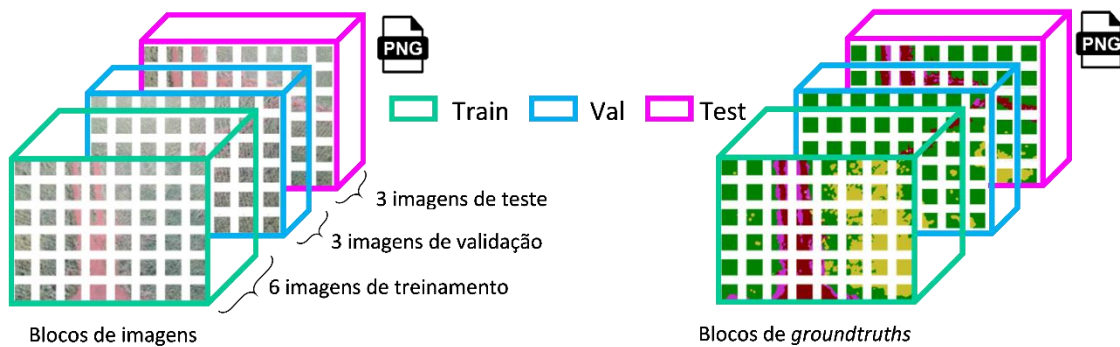
No [Apêndice A.1](#) e [A.2](#) são mostradas miniaturas das 48 imagens originais, além das 12 imagens selecionadas (com seus respectivos *groundtruths*). Se necessário, é possível aumentar o conjunto de dados de treinamento, rotulando-se o restante das imagens originais disponíveis, uma vez que apresentam ângulos e pontos de vista diferentes.



### 5.1.2 Blocos de imagens originais RGB e *groundtruths*

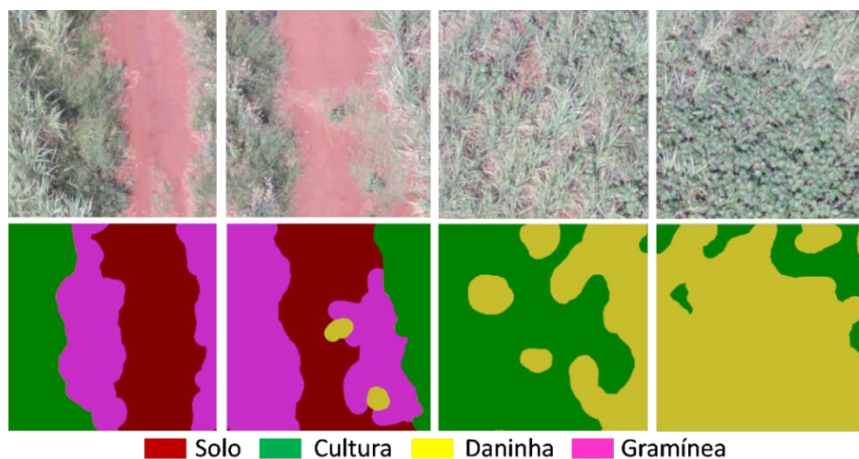
Cada uma das 12 imagens selecionadas de  $4.608 \times 3.456$  pixels e seus *groundtruths* correspondentes foram subdivididos em uma grade regular de  $8 \times 6$  blocos de imagens de  $576 \times 576$  pixels. Escolhemos esse tamanho de bloco padrão para subdividir a imagem por um número inteiro de blocos com aproximadamente o mesmo tamanho da entrada das CNNs, bem como, por abranger uma área contendo uma quantidade suficiente de cana e plantas daninhas. O Apêndice A.2 mostra o conjunto de todos os blocos das imagens selecionadas e seus *groundtruths*.

De acordo com a seleção das imagens na Figura 5.3b, os blocos anotados foram distribuídos no conjunto de treinamento ( $6 * 8 \times 6$  blocos), validação ( $3 * 8 \times 6$  blocos) e teste ( $3 * 8 \times 6$  blocos), como ilustrado na Figura 5.5. O conjunto de treinamento foi usado para otimização dos parâmetros (pesos) das redes, usando um método baseado em gradiente descendente estocástico (SGD). O conjunto de validação foi usado para selecionar os hiperparâmetros das redes, como tamanho de *mini-batch*, taxa de aprendizado, etc. Após o treinamento, o conjunto de teste com três imagens aéreas é usado para predição e avaliação das redes.



**Figura 5.5** – Conjunto de dados particionado em treinamento, validação e teste, composto de imagens originais e seus *groundtruths*, divididos em uma grade regular de  $8 \times 6$  blocos de  $576 \times 576$  pixels.

É importante observar que as classes são desbalanceadas, uma vez que solo, gramínea e daninha apresentam menor número de pixels por imagem aérea, com predominância da classe cultura de cana. A Figura 5.6 mostra em detalhes uma sequência de quatro blocos da imagem 01 da Tabela A.1 e seus *groundtruths* de  $576 \times 576$  pixels.



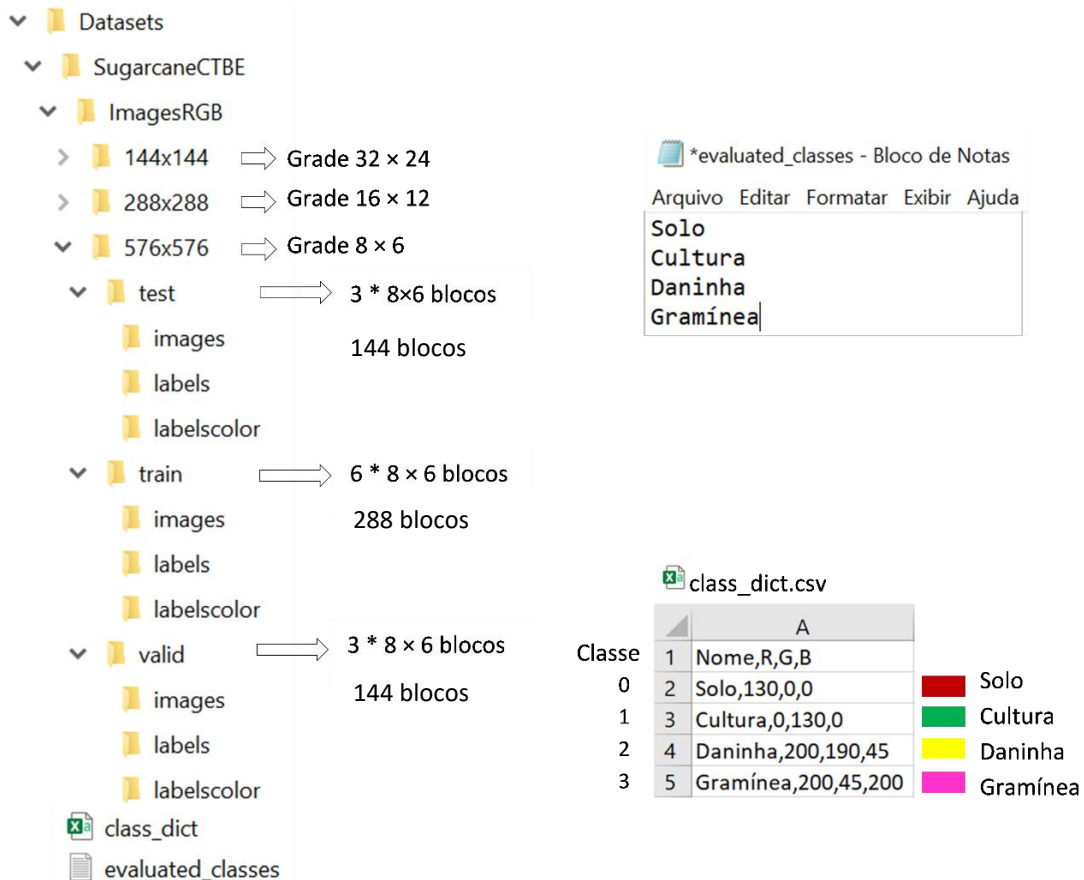
**Figura 5.6** – Detalhes de quatro blocos da imagem 01 e *groundtruths* de  $576 \times 576$  pixels.

**Tabela 5.1** – Conjuntos de dados de treinamento, validação e teste das imagens originais.

Conjunto de dados	Número imagens RGB	Tamanho da imagem (pixels)	GSD médio (cm/pixel)	Grade <sup>10</sup> (linhas × colunas)	Resolução do bloco (pixels)	Número de blocos
Treinamento	6	4.608 × 3.456	1,16	8 × 6	576 × 576	288 (50%)
Validação	3	4.608 × 3.456	1,16	8 × 6	576 × 576	144 (25%)
Teste	3	4.608 × 3.456	1,16	8 × 6	576 × 576	144 (25%)

Os conjuntos de dados de treinamento, validação e teste, listados na [Tabela 5.1](#), – formados pelos blocos de imagens aéreas originais (em formato PNG de 24 bits) e seus *groundtruths* (PNG de 8 bits), além dos *groundtruths* coloridos correspondentes (PNG de 24 bits) –, são armazenados em pastas separadas, como mostra a [Figura 5.7](#). A [Figura A.3](#), no [Apêndice A](#), mostra os blocos das imagens RGB originais, nomeados “*imagefile\_RGB\_LxC.png*”, bem como os blocos de seus *groundtruths* coloridos “*imagefile\_RGB\_LabelColor\_LxC.png*” e de 8 bits “*imagefile\_RGB\_Label8bits\_LxC.png*”, onde *imagefile* é o nome do arquivo da imagem aérea original, listado na [Tabela A.1](#);  $L \times C$  é a posição em linha e coluna de um determinado bloco na grade.

As redes geram mapas de predições com rótulos de classe para cada *pixel*, com valores [0, 1, 2, 3] definidos no arquivo “*evaluated\_classes.txt*”, que são exportados em formato PNG de 8 bits. Além disso, arquivos de predições coloridas correspondentes, em formato PNG de 24 bits, são geradas de acordo com o arquivo “*class\_dict.csv*”, que define os valores das cores RGB de cada classe (separados por vírgulas no Excel), para facilitar a análise visual.

**Figura 5.7** – Hierarquia de pastas do conjunto de dados de treinamento, validação e teste.

<sup>10</sup> Outros tamanhos de grade de blocos foram armazenados em pastas para experimentos adicionais.

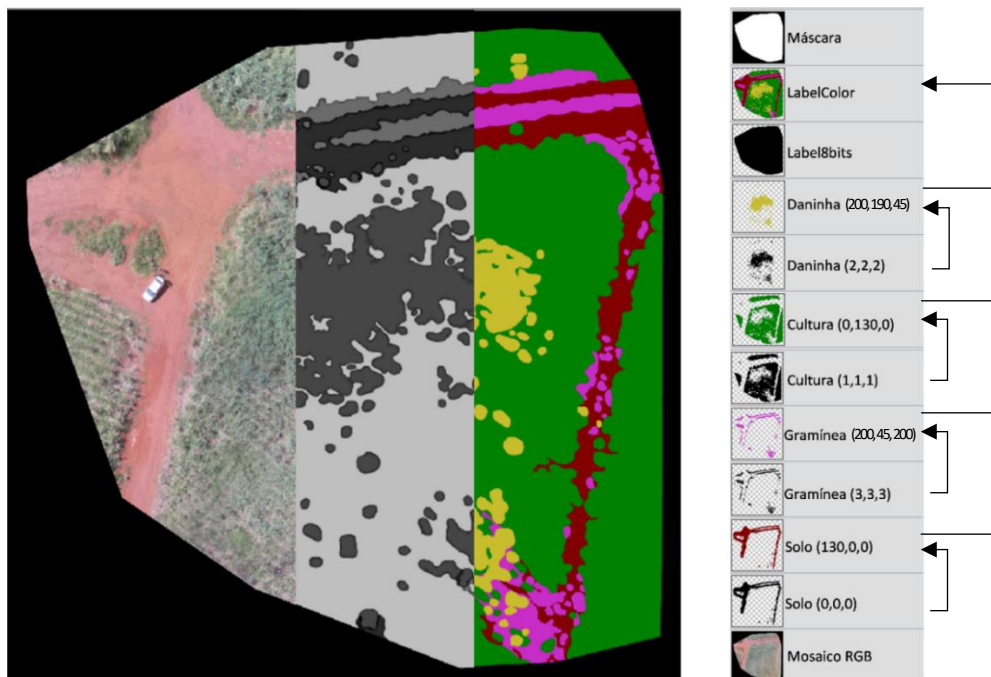


### 5.1.3 Rotulação do ortomosaico RGB

Para gerar o conjunto de dados do ortomosaico RGB para teste das redes de segmentação semântica, primeiro rotulamos manualmente o mosaico RGB em quatro classes, que mostram a distribuição de plantas daninhas do campo experimental de cultivo de cana-de-açúcar. O mapa de reflectância *Red-Edge* é utilizado apenas como referência visual para auxiliar na localização da posição relativa das plantas daninhas, uma vez que o mosaico RGB e os mapas de reflectância não se sobrepõem precisamente, como mostrado na [Figura 4.62](#).

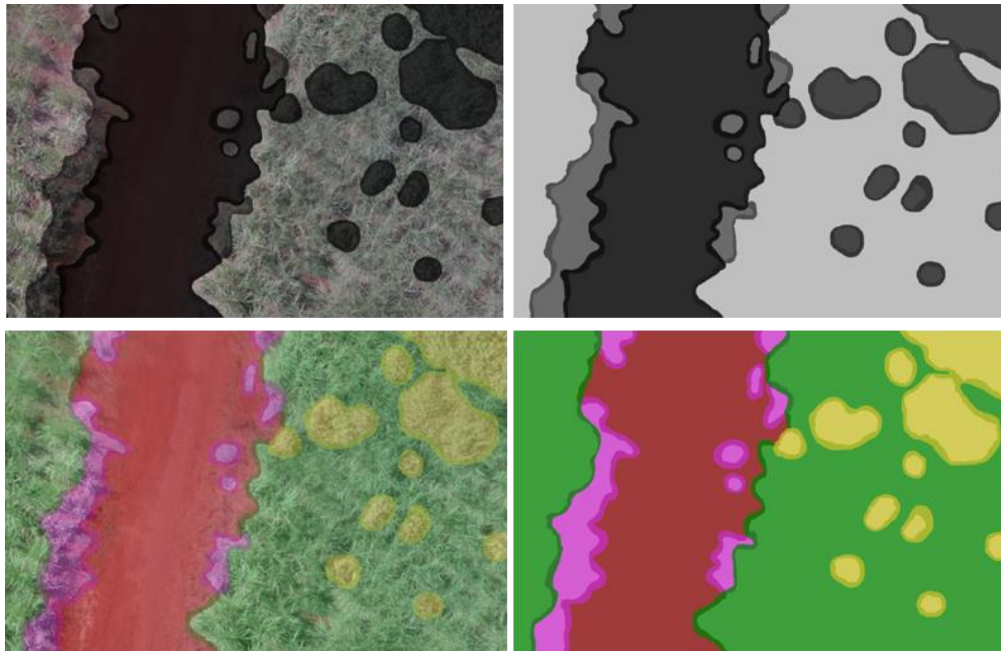
Da mesma forma que nas imagens aéreas, utilizamos o software de edição de imagens Photoshop para rotular cada *pixel* com valores de classe [0, 1, 2, 3]<sup>11</sup>, como mostrado na [Figura 5.8](#). Primeiro, criamos uma camada de 8 bits para cada classe. Por exemplo, a camada “Daninha” tem valores RGB (2, 2, 2) nos *pixels* correspondentes às plantas daninhas e transparente no restante dos *pixels*. Depois, transformamos cada camada de rótulos de 8 bits em uma camada de rótulos de 24 bits colorida correspondente, por exemplo, “Daninha (200, 190, 45)” com *pixels* em amarelo. Durante a edição, podemos variar a transparência de cada camada de 0 a 100%, para visualizar os rótulos sobre o mosaico RGB original, como mostra o recorte na [Figura 5.9](#). Os limites de cada classe foram ligeiramente sobrepostos para não deixar nenhum *pixel* sem rótulo ou com rótulo inválido.

Por fim, as quatro camadas de classes são mescladas para gerar uma camada de *groundtruth* com rótulos de 8 bits (e uma camada com rótulos coloridos de 24 bits) do mosaico completo. Uma camada “Máscara” com a borda preta do mosaico (mostrada na [Figura 4.46](#)) é multiplicada com as camadas “LabelColor” e “Label8bits” para gerar os rótulos de borda com valor 0 (fundo preto). O *groundtruth* do mosaico foi salvo no formato PNG de 8 bits, com valores de classe [0, 1, 2, 3] para cada *pixel*, assim como em formato PNG de 24 bits, com as cores correspondentes das classes (mostrado na [Figura 5.10b](#)).



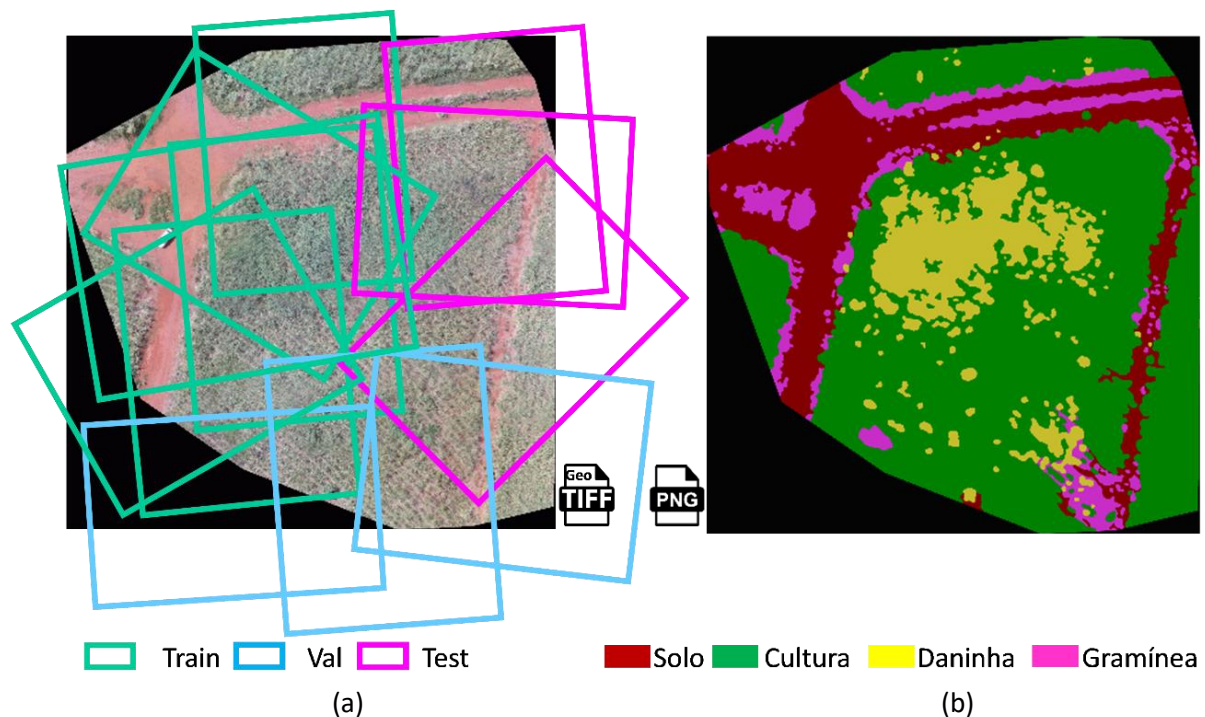
**Figura 5.8** – Rotulação manual dos *pixels* do mosaico, usando software de edição de imagens, para teste das redes de segmentação semântica.

<sup>11</sup> Os *pixels* pretos nas bordas do mosaico também têm valor 0, mas não correspondem à uma nova classe “borda” (que não foi treinada pela rede), nem à classe “solo”, por isso, os blocos com *pixels* pretos devem ser descartados para extrair medidas de desempenho (ou para treinamento direto com blocos do mosaico).



**Figura 5.9** – *Groundtruths de 8 e 24 bits com 4 camadas transparentes sobre o mosaico.*

Apesar da área mapeada no mosaico ser a mesma capturada pelas imagens aéreas originais utilizadas no treinamento, validação e teste (como mostra a [Figura 5.10a](#)), para efeito de teste das redes de segmentação, consideramos o mosaico um conjunto de dados independente, uma vez que as imagens aéreas têm diferentes ângulos de rotação, e os *pixels* do ortomosaico RGB apresentam características diferentes das imagens aéreas originais de treinamento (como já mostrado na [Figura 4.48](#)). Com isso, os atributos extraídos pelas redes convolucionais durante o treinamento com as imagens originais podem não corresponder precisamente aos atributos extraídos do mosaico durante o teste.



**Figura 5.10** – *Sobreposição das imagens aéreas no mosaico (a); groundtruth colorido de 24 bits do mosaico RGB com  $8.196 \times 8.260$  pixels.*

### 5.1.4 Blocos do ortomosaico RGB e *groundtruth*

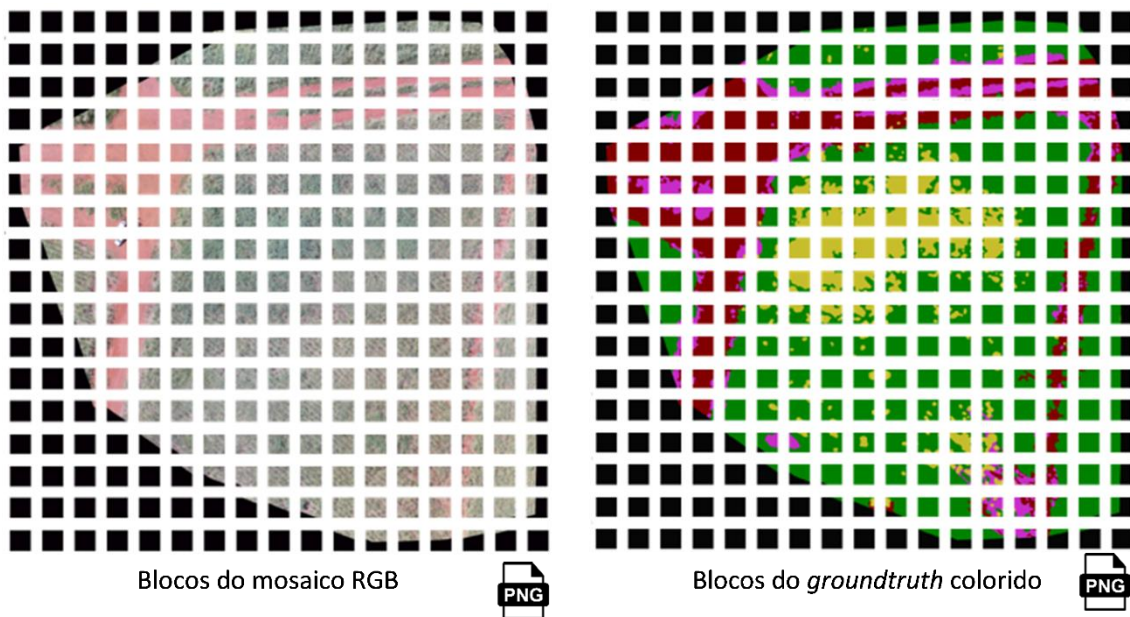
A [Tabela 5.2](#) apresenta os detalhes do conjunto de teste para avaliação das redes de segmentação semântica a partir do mosaico RGB.

**Tabela 5.2** – *Conjunto de dados de teste sobre mosaico RGB.*

Descrição	Mosaico RGB	Unidade
Resolução espacial	8.196 × 8.260	pixels
GSD médio	1,16	cm/pixel
Grade (linhas x colunas)	17 × 17	blocos
Resolução do bloco	512 × 512	pixels
Preenchimento (col x lin)	508   444	pixels
#blocos efetivos	#blocos - 100	blocos

O ortomosaico RGB em formato original GeoTIFF tem resolução espacial de  $8.196 \times 8.260$  pixels, com um GSD médio de 1,16 cm/pixel, que captura detalhes de alta resolução da vegetação. O mosaico RGB e o *groundtruth* foram expandidos para  $8.704 \times 8.704$  pixels, com preenchimento de pixels pretos (zero) nas bordas.

A [Figura A.6](#), no [Apêndice A](#), mostra o mosaico RGB expandido nas bordas, nomeado “*mosaic\_RGB\_padding.png*”, com seu *groundtruth* colorido de 8 e 24 bits. O preenchimento nas bordas permite que o mosaico e os *groundtruths* sejam particionados em uma grade regular de  $17 \times 17$  blocos de  $512 \times 512$  pixels, compondo o conjunto de teste das redes, como mostrado na [Figura 5.11](#) (visualizadas em tamanho maior nas [Figuras A.4 e A.5](#)).



**Figura 5.11** – *Conjunto de dados de teste: ortomosaico RGB, anotado e expandido para  $8.704 \times 8.704$  pixels, particionado em  $17 \times 17$  blocos de  $512 \times 512$  pixels, para teste final das redes de segmentação semântica.*

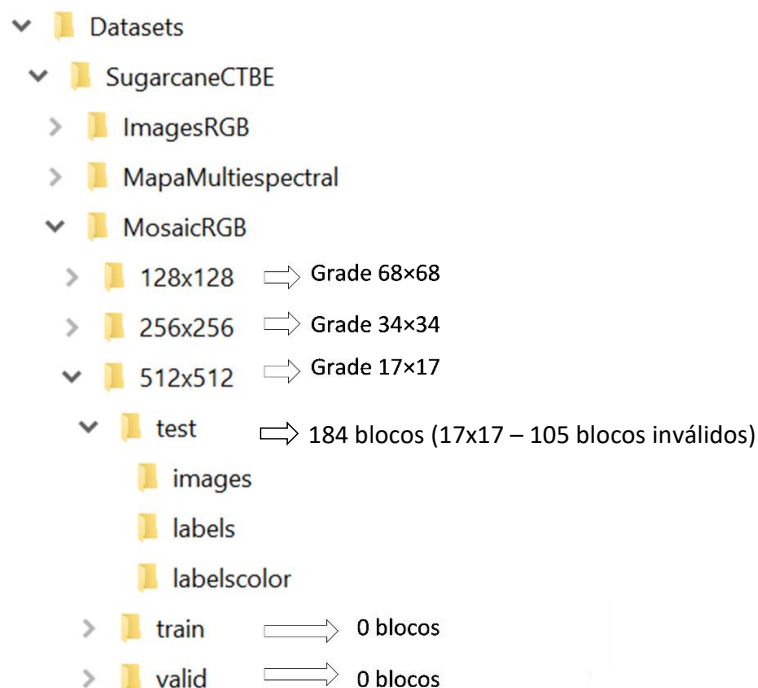


Na [Tabela 5.2](#), as informações de preenchimento indicam o número de *pixels* pretos adicionados nas bordas do mosaico, com objetivo de obter um tamanho de mapa ortomosaico divisível por um número inteiro de blocos. Por exemplo, anexamos 508 colunas e 444 linhas de *pixels* pretos nas bordas verticais e horizontais do mosaico, obtendo um mapa expandido de  $8.704 \times 8.704$  *pixels*. Assim, conseguimos uma grade regular sobre o mosaico, consistindo de 17 blocos por linha ( $17 * 512$  *pixels*) e 17 blocos por coluna ( $17 * 512$  *pixels*). Outros tamanhos de grade de blocos foram gerados para testes adicionais.

O número de blocos efetivos é o número de blocos que contêm valores de *pixels* válidos para medidas de desempenho, isto é, excluindo os blocos que contêm *pixels* pretos (sem classe definida) localizados nas bordas do ortomosaico, uma vez que não foram treinados pelas redes. As [Figuras A.4 e A.5](#) mostram que o mosaico RGB tem 105 blocos de  $512 \times 512$  *pixels* não válidos; portanto, estes não são usados para medidas de desempenho, uma vez que os *pixels* preditos nas bordas do mosaico são imprevisíveis. Em outras palavras, os blocos preditos inválidos de 8 bits não podem ser comparados com os blocos de *groundtruths* correspondentes para obter uma métrica de desempenho.

Os blocos do mosaico RGB de teste, nomeados “*mosaic\_RGB\_padding\_LxC.png*”, com seus *groundtruths* correspondentes de 8 e 24 bits, onde *L* é a linha e *C* a coluna de cada bloco na grade, foram salvos nas subpastas “*images*”, “*labels*” e “*labelscolor*”, respectivamente (como mostrado na [Figura 5.12](#)). As pastas “*train*” e “*valid*” não têm nenhum bloco, mas poderiam ser usadas para colocar blocos de uma área parcial do mosaico (exceto blocos com bordas pretas) para executar o treinamento e validação diretamente a partir do mosaico.

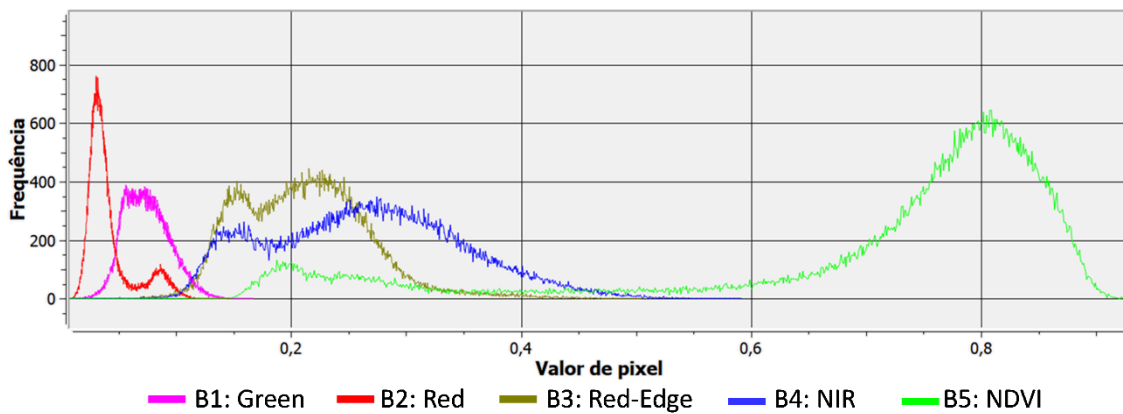
Para teste de cada rede de segmentação, o conjunto de blocos de teste do mosaico serve como entrada da rede, mas seus blocos *groundtruths* de 8 bits correspondentes não são fornecidos à rede na etapa de predição. Desta forma, os *groundtruths* de 8 bits são usados apenas para medição do desempenho da rede após a predição, comparando-os com os blocos preditos de 8 bits.



**Figura 5.12** – Hierarquia de pastas do conjunto de teste do mosaico RGB.

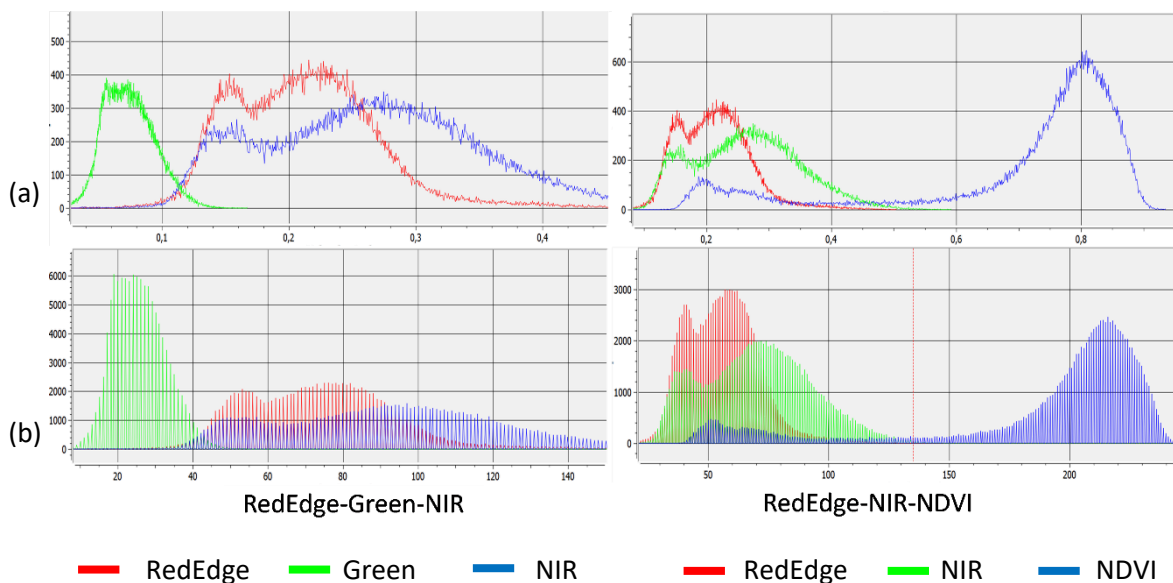
### 5.1.5 Rotulação de mosaico multiespectral multicanal

Para treinar e testar as redes de segmentação semântica em dados multiespectrais, primeiro geramos mosaicos multicanais com diferentes combinações de mapas de reflectância e mapa de índice de vegetação NDVI. Na [Figura 5.13](#), analisamos o histograma dos quatro mapas de reflectância *Green*, *Red*, *Red-Edge* e NIR ([Figura 4.49](#)) e do mapa de índice de vegetação NDVI ([Figura 4.51](#)), com valores de *pixel* normalizados entre 0 e 1. Com isso, decidimos empilhar os mapas *RedEdge–Green–NIR* e *RedEdge–NIR–NDVI* nos três canais RGB, usando o QGIS para gerar composições coloridas multicanais (como discutido na [Seção 4.4.5](#)).



**Figura 5.13** – Histograma dos mapas de reflectância multiespectrais e mapa NDVI.

Cada mosaico multicanal *RedEdge–Green–NIR* e *RedEdge–NIR–NDVI*, visualizado como composição colorida na [Figura 5.15](#), preserva o formato GeoTIFF com informações de georreferenciamento, resolução espacial de  $2.470 \times 2.803$  *pixels* e GSD médio de 4,63 *cm/pixel*, tendo valores de reflectância em ponto flutuante, como mostrado nos histogramas da [Figura 5.14a](#). No entanto, os arquivos PNG de 24 bits correspondentes ([Figura 5.16](#)) têm valores de intensidade de *pixel* de 0 a 255 para cada canal, sem informação geográfica<sup>12</sup>, com histogramas mostrados na [Figura 5.14b](#).

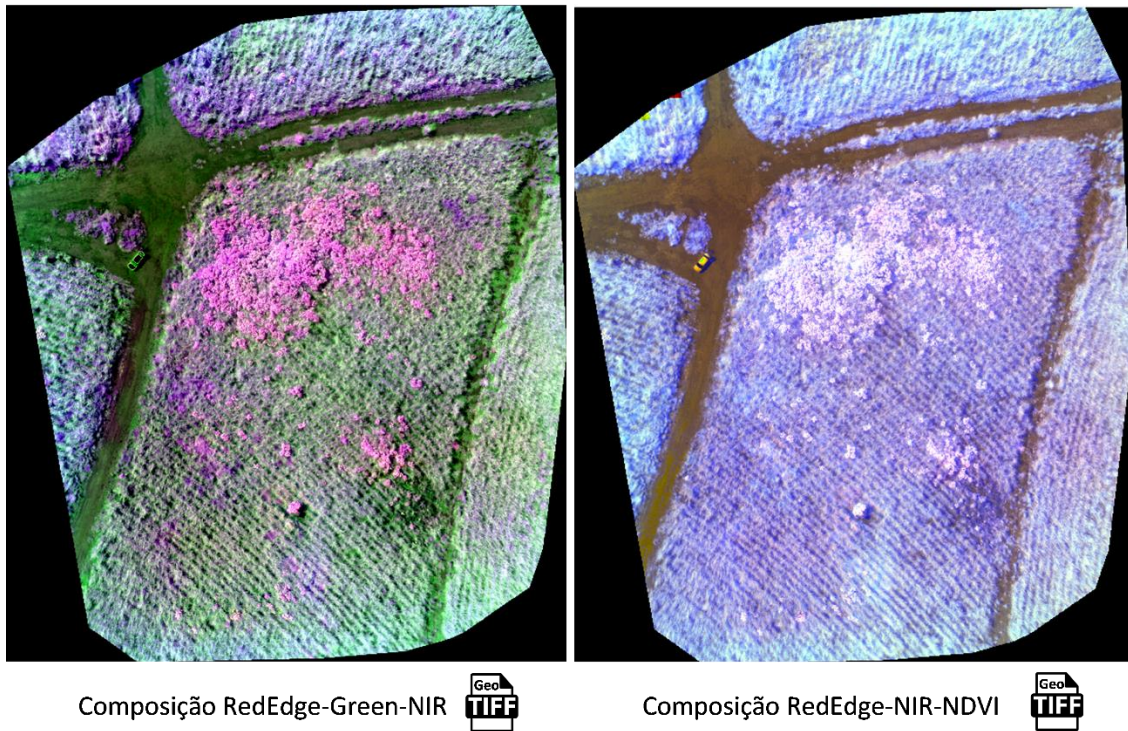


**Figura 5.14** – Histogramas dos mosaicos multiespectrais multicanais GeoTIFF em ponto flutuante (a) e convertido para PNG de 24 bits (b).

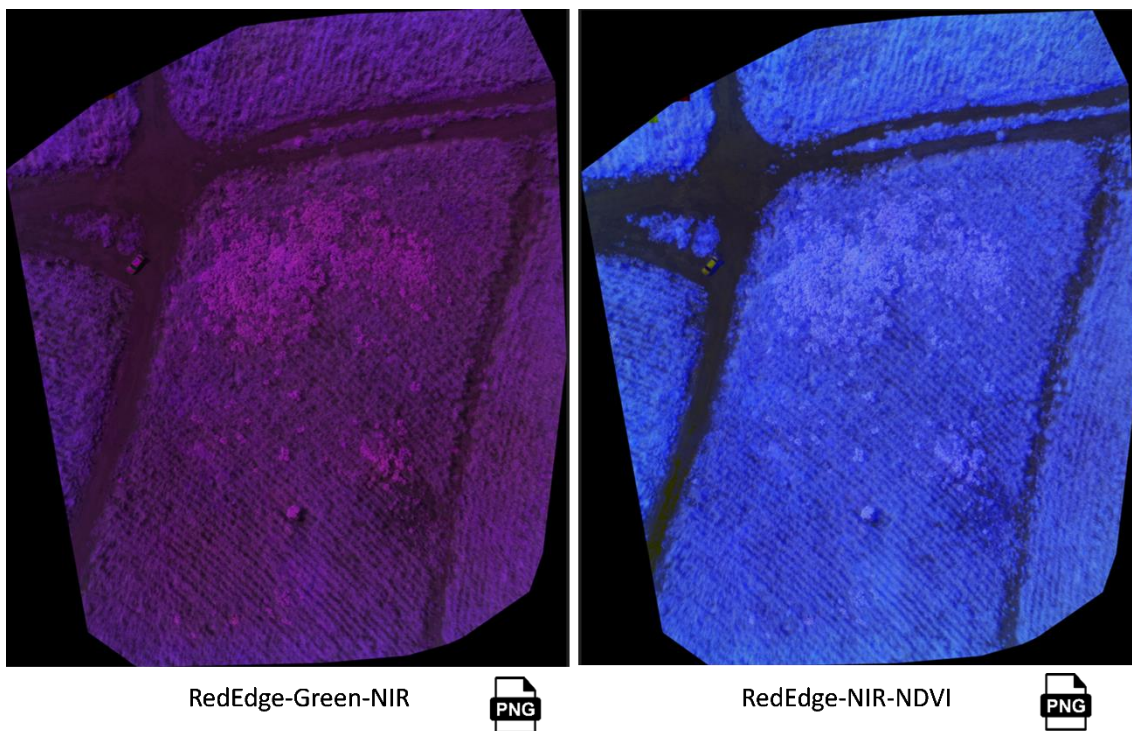
<sup>12</sup> O programa de referência da rede neural de segmentação semântica normaliza os valores de *pixel* dividindo-os por 255 na entrada, portanto, não utilizamos o arquivo original GeoTIFF com valores normalizados entre 0 e 1.



As composições coloridas destacam as plantas daninhas da vegetação, como mostra a [Figura 5.15](#).

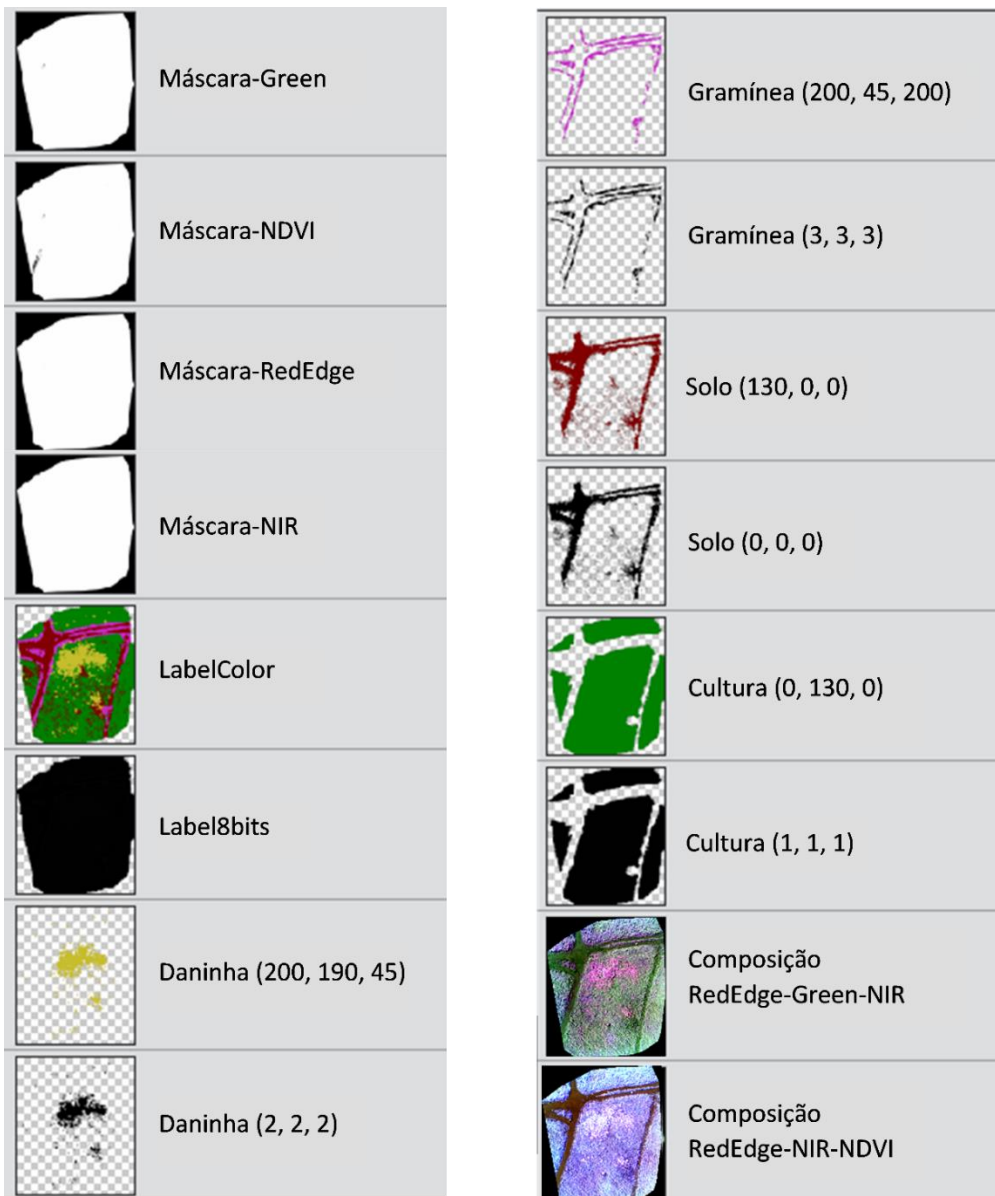


**Figura 5.15** – Mosaicos multiespectrais multicanais em formato *GeoTIFF* de três canais (tipo float de 32 bits), visualizados como composição colorida no QGIS.



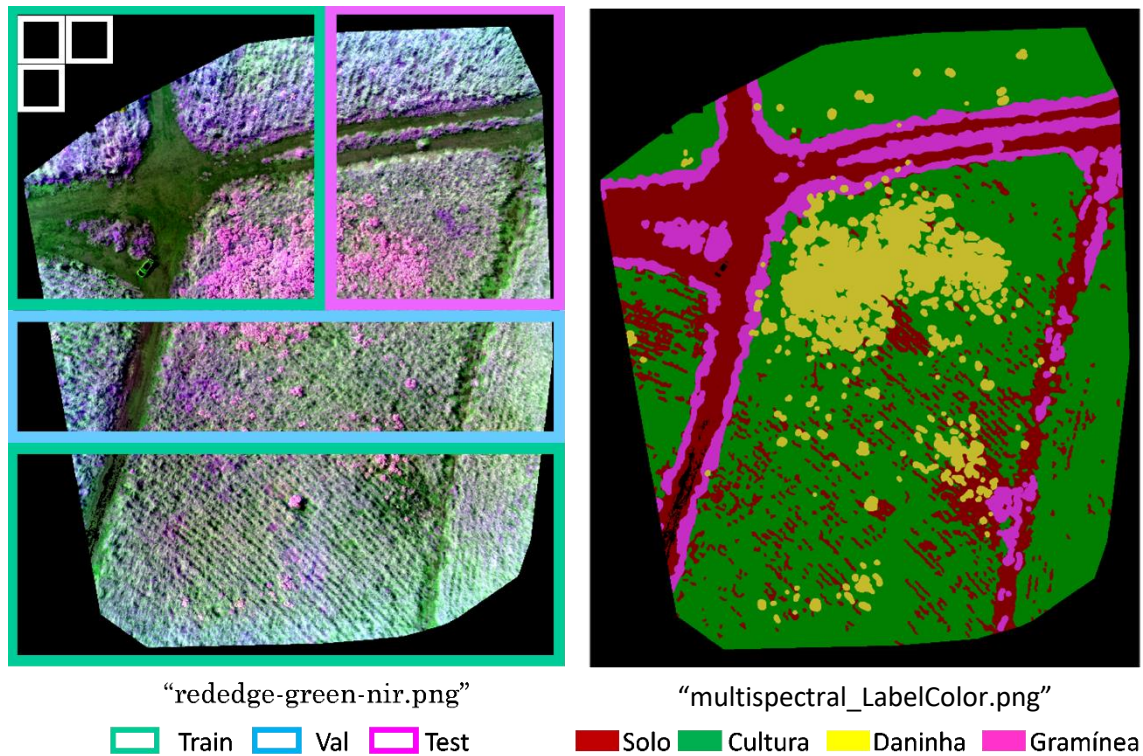
**Figura 5.16** – Mosaicos multiespectrais em formato *PNG* com três canais de 8 bits.

Um único arquivo de *groundtruth* de 8 e 24 bits cada foi gerado para as duas composições multiespectrais, seguindo o mesmo procedimento de rotulação manual no Photoshop (como descrito na [Seção 5.1.3](#)), com camadas separadas de rótulos para cada classe e camadas mescladas das quatro classes, como mostra a [Figura 5.17](#). No final, cada camada – “*Labels8bits*” e “*LabelsColor*” – é multiplicada por três máscaras dos mapas de reflectância ([Figura 4.49](#)) que fazem parte da composição *RedEdge–Green–NIR* ou *RedEdge–NIR–NDVI*, para gerar o arquivo de *groundtruth* final com bordas pretas zeradas ([Figura 5.18](#)). O [Apêndice A.4](#) mostra as composições multiespectrais e o *groundtruth*.



**Figura 5.17** – Rotulação manual dos mapas multiespectrais multicanais.





**Figura 5.18** – Mosaico multispectral multicanal e seu groundtruth colorido de  $2.560 \times 3.072$  pixels, com preenchimento nas bordas, particionados em  $10 \times 12$  blocos de  $256 \times 256$  pixels, para treinamento, validação e teste das redes de segmentação semântica.

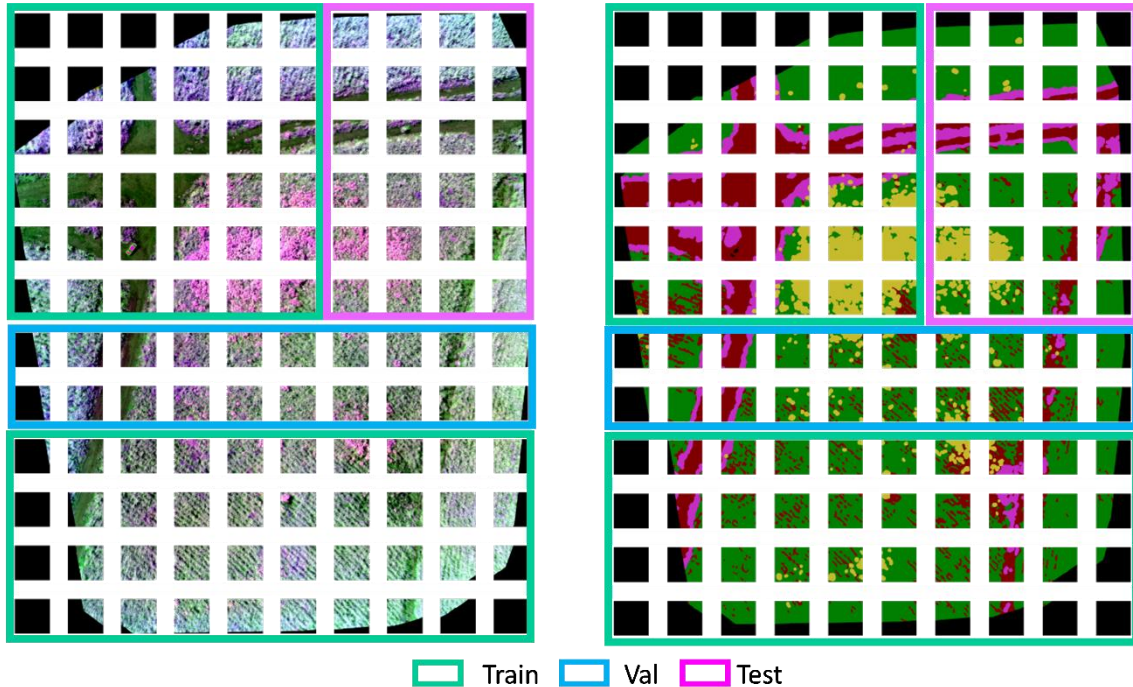
A [Tabela 5.3](#) apresenta os detalhes do conjunto de dados a partir dos mosaicos multispectrais multicanais. Primeiro, os mosaicos foram expandidos para  $2.560 \times 3.072$  pixels com preenchimento de zeros nas bordas e salvos em formato PNG de 24 bits.

Os mosaicos *RedEdge–Green–NIR* e *RedEdge–NIR–NDVI*, assim como o *groundtruth* expandido são particionados em  $10 \times 12$  blocos de  $256 \times 256$  pixels. A [Figura A.8](#) mostra 45 blocos inválidos (com fundo preto) na grade de  $10 \times 12$  blocos. Os blocos efetivos (excluindo os blocos inválidos) podem ser usados para cálculo de métricas de desempenho.

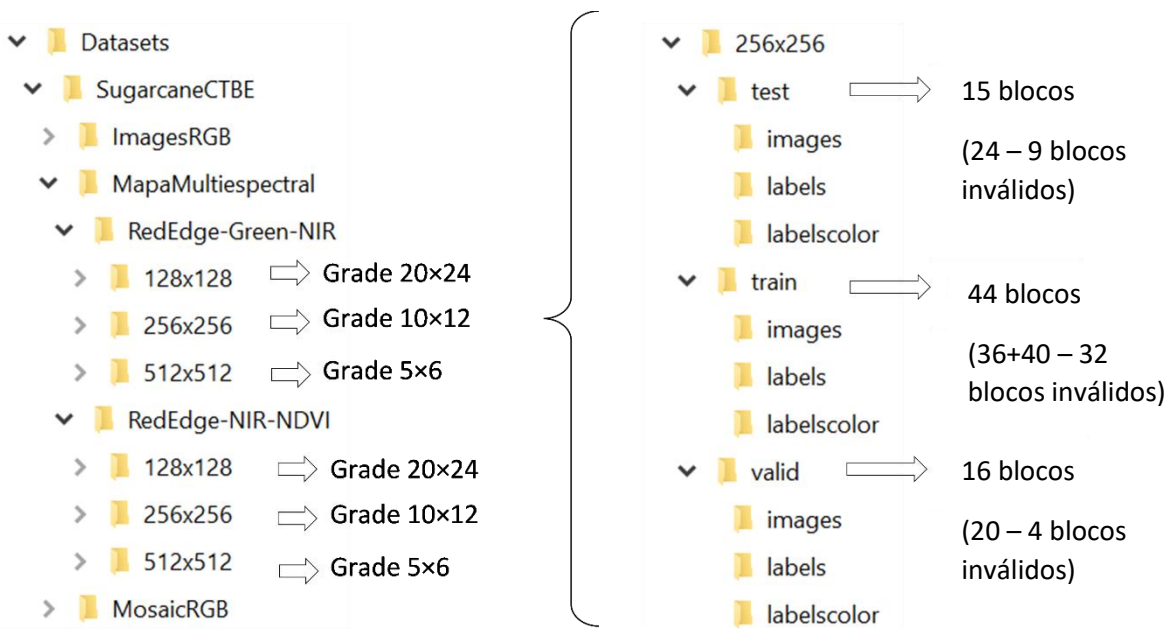
**Tabela 5.3** – Conjunto de dados com blocos do mapa multicanal.

Descrição	Mosaico multicanal multispectral	Unidade
Resolução espacial	$2.470 \times 2.903$	pixels
GSD médio	4,63	cm/pixel
Combinação de canais	RedEdge-Green-NIR RedEdge-NIR-NDVI	
#blocos (linhas x colunas)	$10 \times 12$	blocos
Resolução do bloco	$256 \times 256$	pixels
Preenchimento (col x lin)	90   169	pixels
#blocos efetivos	#blocos - 30	blocos

O conjunto total de blocos é distribuído em subconjuntos de treinamento, validação e teste, com 44, 16 e 15 blocos (sem blocos inválidos), respectivamente, de acordo com as subáreas disjuntas selecionadas do mosaico, como mostrado na [Figura 5.19](#). Os arquivos de cada bloco em formato PNG foram organizados e armazenados em pastas, como mostra a [Figura 5.20](#). Outros tamanhos de grade de blocos foram gerados para testes adicionais.



**Figura 5.19** – Blocos do conjunto de treinamento, validação e teste.



**Figura 5.20** – Hierarquia de pastas do conjunto de teste dos mapas multiespectrais.

Finalizada a primeira etapa do fluxo de trabalho, seguimos com a segunda etapa para realizar experimentos com as nove redes neurais selecionadas, treinadas e testadas na tarefa de segmentação semântica em nosso conjunto de dados.

## 5.2 Experimentos com redes de segmentação semântica

Nesta seção, testamos nove arquiteturas de redes de segmentação semântica de imagens bem-sucedidas em conjuntos de dados de referência, como PASCAL-VOC, que foram treinadas e ajustadas em nosso conjunto de dados.

### 5.2.1 Implementação das redes de segmentação semântica

Para realizar a avaliação de desempenho das redes selecionadas, utilizamos um código-fonte de referência dos modelos de segmentação semântica – implementado por Yang Lu – disponível publicamente no GitHub [LU 2019]. Este código foi escrito em linguagem Python, utilizando a biblioteca Keras [CHOLLET 2018] com TensorFlow [ABADI *et al.* 2015] – uma biblioteca de código aberto para desenvolvimento de redes neurais, que dá suporte para execução em vários dispositivos, incluindo GPU. A biblioteca Keras disponibiliza os modelos CNN utilizados como *backbone* das redes de segmentação semântica (inclusive com pesos pré-treinados em conjuntos de dados de referência). No entanto, Yang Lu reescreveu os códigos das CNN-base, como discutido no Capítulo 2, a fim de usar a convolução dilatada nos dois últimos blocos, sem operação de subamostragem dos mapas de atributos (para manter a resolução espacial nas camadas finais da CNN), o que impossibilita o uso do código da CNN original em Keras.

O código de referência suporta os seguintes modelos de segmentação semântica: FCN-32/16/8s [LONG *et al.* 2015]; U-Net [RONNEBERGER *et al.* 2015]; SegNet [BADRINARAYANAN *et al.* 2017] e Bayesian SegNet [KENDALL *et al.* 2015]; PSPNet [ZHAO *et al.* 2017]; RefineNet [LIN, MILAN *et al.* 2017]; PAN [LI *et al.* 2018]; DeepLab-v3 [CHEN *et al.* 2017] e DeepLab-v3+ [CHEN *et al.* 2018b]; DenseASPP [YANG *et al.* 2018] e BiSeNet [YU *et al.* 2018]. Os modelos CNN-base suportados por cada rede são: VGG-16/19 [SIMONYAN e ZISSERMAN 2015], ResNet-50/101/152 [HE *et al.* 2016a], DenseNet-121/169/201/264 [HUANG *et al.* 2017], MobileNet-v1 [HOWARD *et al.* 2017], MobileNet-v2 [SANDLER *et al.* 2018], Xception [CHOLLET 2017] e Xception-DeepLab [CHEN *et al.* 2018b].

Para avaliação das redes em nosso conjunto de dados, limitamos os experimentos em nove modelos de segmentação semântica, com *backbones* listados na Tabela 5.4. Adaptamos o código-fonte de referência destas redes (implementado com CNNs inicializadas aleatoriamente), para permitir a transferência de aprendizado com pesos das CNNs pré-treinadas disponibilizados pelo Keras. No Apêndice B, são mostrados os modelos das redes de segmentação, tal como implementados em nosso código adaptado, de acordo com as informações apresentadas nas Seções 2.3 e 2.4 do Capítulo 2.

**Tabela 5.4** – Modelos de redes de segmentação semântica com diferentes *backbones*.

Modelo	Modelo-base (CNN)	Parâmetros
FCN-8s	VGG-16	134.284.632
U-Net	VGG-16	25.866.388
SegNet	VGG-16	29.442.260
RefineNet	ResNet-50	58.797.444
BiSeNet	Xception	26.081.488
PSPNet	ResNet-50	46.671.252
DeepLab-v3	ResNet-50	39.123.332
DeepLab-v3+	ResNet-50	40.373.908
DenseASPP	DenseNet-121	9.284.484



### 5.2.2 Estratégias de treinamento

Em nossos experimentos, todas as redes foram adaptadas à nossa tarefa específica de segmentação semântica de plantas daninhas, com treinamento a partir do conjunto de dados de imagens aéreas originais RGB de um campo experimental de cultura de cana-de-açúcar. Conforme discutido na [Seção 2.5](#), em muitas aplicações pode não ser viável treinar totalmente uma CNN em um conjunto de dados específico, pois geralmente requer uma quantidade considerável de dados rotulados, muitas vezes não disponível, para evitar sobreajuste da rede. Para amenizar este problema, utilizamos CNNs pré-treinadas no conjunto de dados ImageNet [[DENG et al. 2009](#)], disponíveis como parte da biblioteca Keras, tais como: VGG-16 [[SIMONYAN e ZISSERMAN 2015](#)], Xception [[CHOLLET 2017](#)], ResNet-50 [[HE et al. 2016a](#)] e DenseNet-121 [[HUANG et al. 2017](#)]. Para isso, empregamos as três primeiras de quatro estratégias de treinamento, em nosso conjunto de dados:

- (1) Inicialização aleatória: treinamento de todas as camadas da rede de segmentação com inicialização aleatória de pesos, para obter um ponto referencial de desempenho – denominada como A (pesos aleatórios) nas tabelas de resultados;
- (2) transferência de aprendizado com reajuste fino: inicialização da CNN-base com pesos pré-treinados no conjunto de dados ImageNet [[DENG et al. 2009](#)], reajustando todos os pesos da rede em nosso conjunto de dados – denominada AT (*All Trainable*);
- (3) transferência de aprendizado sem reajuste de pesos: inicialização da CNN-base com pesos pré-treinados, congelando todos os blocos transferidos (para usar a CNN como extrator de atributos); treinamento apenas das camadas restantes da rede de segmentação – denominada NT (*No Trainable*);
- (4) transferência de aprendizado com ajuste fino dos últimos blocos: inicialização da CNN-base com pesos pré-treinados, congelando os três primeiros blocos convolucionais da CNN, com ajuste fino dos dois últimos blocos transferidos e treinamento das camadas restantes da rede de segmentação – denominada UT (*Upper Trainable*).

Conforme apresentado na [Figura 2.123](#) na [Seção 2.5](#), dependendo do modelo de rede de segmentação e do *backbone* pré-treinado empregado (com ou sem modificação nos últimos blocos da CNN), se nenhum bloco da CNN for congelado durante o treinamento, todos os pesos do *backbone* são atualizados (A ou AT); se todos os blocos transferidos forem congelados, a rede-base pré-treinada no ImageNet é usada como extrator de atributos para uma rede de segmentação (NT). Para melhor adaptar a rede-base pré-treinada na tarefa específica (sem aumentar demais o número de parâmetros e, conseqüentemente, a possibilidade de sobreajuste), as camadas inferiores podem ser congeladas para realizar o ajuste fino apenas nas últimas camadas do *backbone* (UT); porém, como os resultados são inferiores ao AT, como mostrado na [Figura 2.124](#), não utilizamos esta estratégia.

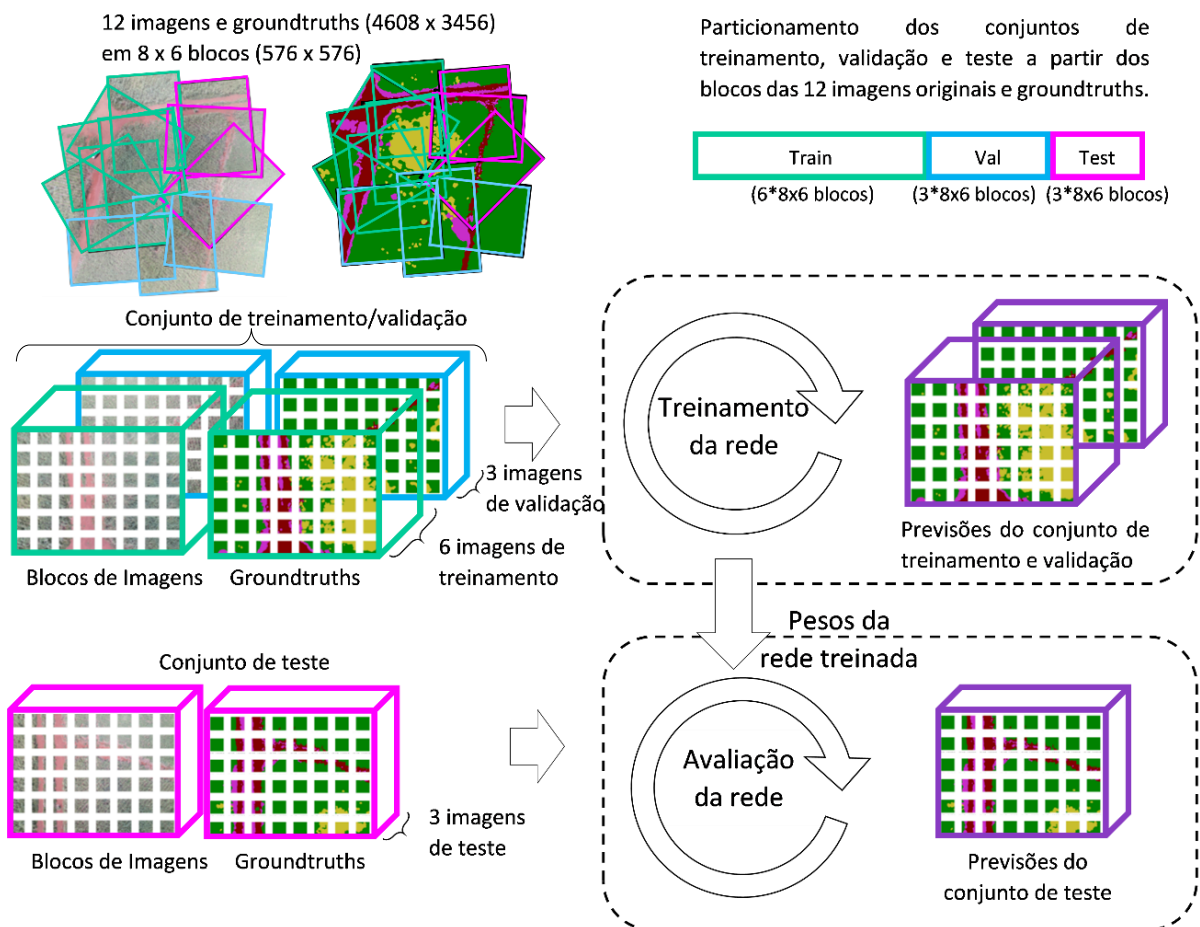
Além disso, utilizamos a estratégia de aumento artificial do conjunto de dados de treinamento, denominado DA (*Data Augmentation*), para diminuir o problema de sobreajuste da rede – essencial quando poucos dados de treinamento estão disponíveis [[RUSSAKOVSKY et al. 2015](#)]. No caso de imagens aéreas, aplicamos as transformações nos blocos de imagens de entrada durante o treinamento, tais como: variações de rotação, espelhamento horizontal e vertical, além de variação de iluminação (intensidade de brilho), simulando as condições reais de voo durante o mapeamento aéreo em campo.

A seguir, descrevemos os procedimentos para realização dos experimentos, considerando que nosso objetivo não é maximizar o desempenho absoluto, procurando o conjunto ideal de hiperparâmetros de cada rede, mas apenas comparar os resultados das estratégias de transferência e avaliar o desempenho das diferentes arquiteturas.

### 5.2.3 Treinamento/validação e teste das redes de segmentação

Nesta seção, descrevemos os procedimentos para execução do código de treinamento/validação e do código de teste das redes selecionadas. Todos os experimentos foram realizados em uma GPU NVIDIA® GeForce GTX TITAN X com 12GB de memória RAM (CUDA 11.4) ou NVIDIA™ RTX A5000 com 24 GB de memória RAM (CUDA 11.6).

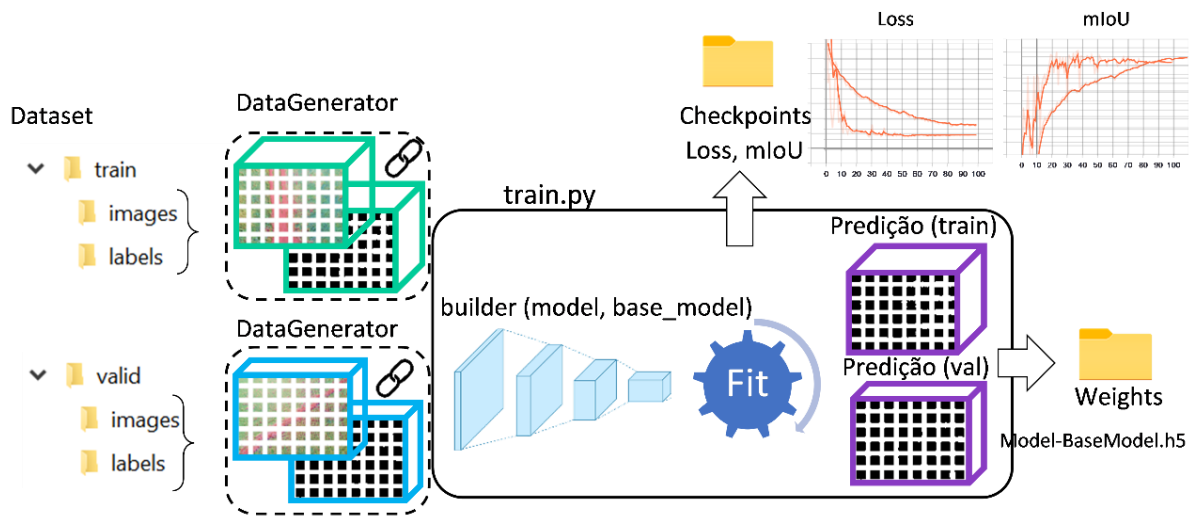
Conforme ilustrado na [Figura 5.21](#), o conjunto de dados de treinamento, validação e teste é composto de [6, 3, 3] imagens aéreas originais rotuladas, particionadas em uma grade regular de  $8 \times 6$  blocos de  $576 \times 576$  pixels, distribuídos em uma proporção de 50%, 25% e 25%, respectivamente. Os conjuntos de treinamento e validação são usados para ajustar os parâmetros (pesos) da rede, minimizando o erro de previsão. O conjunto de teste é usado para avaliar o desempenho da rede treinada.



**Figura 5.21** – Esquema de treinamento/validação e teste das redes de segmentação, a partir da grade de  $8 \times 6$  blocos de  $576 \times 576$  pixels das imagens originais RGB e seus groundtruths correspondentes.

### 5.2.3.1 Treinamento/validação com blocos de imagens RGB

Primeiro, executamos o processo de treinamento e validação dos modelos de rede selecionados, como mostra a [Figura 5.22](#).



**Figura 5.22** – Processo de treinamento/validação dos modelos de segmentação semântica.

Para cada treinamento, executamos o código “*train.py*”, definindo os parâmetros principais: *model* (modelo de segmentação), *base\_model* (modelo *backbone*), *num\_classes* e *dataset* (caminho do conjunto de dados de treinamento e validação).

```
>> python train.py --model MODEL --base_model BASE_MODEL --num_classes NCLASSES --dataset DATASET ...
```

Os parâmetros completos da execução do programa de treinamento são:

model	MODEL = {FCN-8s, UNet, SegNet, RefineNet, PSPNet, DenseASPP, DeepLabV3, DeepLabV3Plus, BiSegNet}	
base_model	BASE_MODEL = {VGG16, ResNet50, DenseNet121, Xception, Xception-DeepLab}	
dataset	DATASET = Datasets/SugarcaneCTBE/ImagesRGB/576x576	
num_classes	NCLASSES = 4	Número de classes
crop_height	CROP_HEIGHT = 256	Redimensionamento da altura da imagem
crop_width	CROP_WIDTH = 256	Redimensionamento da largura da imagem
batch_size	BATCH_SIZE = 8	Tamanho do lote de imagens
num_epochs	NUM_EPOCHS = 100	Número de épocas de treinamento
num_valid_images	NVALID_IMAGES = 144	Número de imagens no conjunto de validação
loss	LOSS = {ce}	Função de perda “ <i>categorical cross-entropy</i> ”
optimizer	OPTIMIZER = {Adam}	Algoritmo de aprendizado por gradiente
learning_rate	LEARNING_RATE = 3e-4	Taxa de aprendizado
h_flip	H_FLIP = {True, False}	} Aumento artificial de dados
v_flip	V_FLIP = {True, False}	
brightness	BRIGHTNESS 0.8 1.2	
rotation	ROTATION = 90.0	
data_aug_rate	DATA_AUG_RATE = 0.5	

Todos os modelos foram treinados usando o mesmo protocolo fixo de treinamento, com função de perda “*categorical cross-entropy*” [ZHANG e SABUNCU 2018] e função de otimização Adam [KINGMA e BA 2014].

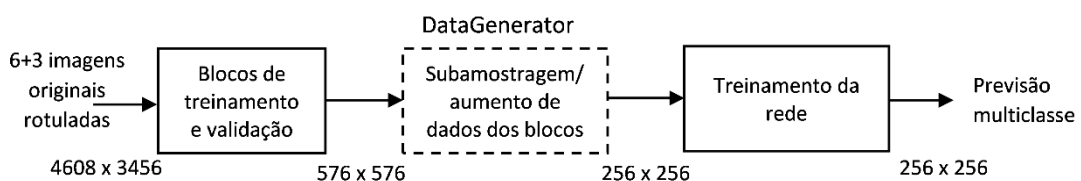
A rede de segmentação semântica prevê uma saída densa de 8 bits com probabilidade multiclasse (solo, cultura, daninha e gramínea) para cada bloco de treinamento, comparando-a com seu *groundtruth* de 8 bits, para medir a função de perda ou erro de treinamento. A função de perda “*categorical cross-entropy*” calcula o erro médio de previsão para cada minilote de blocos de imagens de treinamento. Depois, o gradiente da função de custo é calculado em relação a todos os pesos da rede, usando a regra da cadeia para obter as derivadas parciais e, então, os parâmetros da rede são atualizados pelo método gradiente descendente. O algoritmo de otimização, Adam, baseado no algoritmo gradiente descendente estocástico, ajusta os pesos para minimizar a função de perda. A taxa inicial de aprendizado (*learning rate*) foi definida para 0,0003. O algoritmo de aprendizado executa até atingir o número total de épocas de treinamento.

O número de épocas para atualização de pesos (ou seja, o número de vezes que a rede vê o conjunto inteiro de treinamento) foi definido para 100 e o tamanho do *mini-batch* é 8. Um valor pequeno de lote aumenta o tempo de treinamento, mas um lote maior leva a uma perda no desempenho da generalização [LECUN *et al.* 2012]. Isto é chamado de *generalization gap*, ou seja, o desempenho do modelo em conjuntos de dados de teste geralmente é pior quando treinado com lotes grandes em comparação com lotes pequenos, segundo [KESKAR *et al.* 2016]. Para diminuir o tempo, podemos treinar por um número menor de épocas, mas a rede tende a ter mais problemas de generalização.

Normalmente, as redes são treinadas sem subamostragem na entrada, isto é, no nosso caso, com o conjunto de treinamento formado por blocos rotulados de imagens aéreas de  $576 \times 576$  *pixels*. Com a finalidade de diminuir os custos computacionais e tempo de processamento, todos os blocos com GSD médio de 1,16 *cm/pixel* (correspondente a um voo de 40 metros de altura) foram subamostrados para gerar uma entrada de  $256 \times 256$  *pixels*, simulando um GSD de  $2 \times 1,16$  *cm/pixel* (que equivale a uma altitude de voo de 60 metros).

Para aumentar a capacidade de aprendizado e generalização do modelo, o conjunto de dados de treinamento foi aumentado artificialmente com transformações nos blocos rotulados – variando a intensidade de brilho, rotação e reflexão horizontal/vertical (*flip*), para consequentemente reduzir o sobreajuste da rede, conforme discutido na Seção 2.1.2. O conjunto de validação não é aumentado.

A Figura 5.23 mostra o módulo interno *DataGenerator*, com operações opcionais de redimensionamento e aumento de dados, que prepara efetivamente os dados – com conjunto aumentado de blocos de imagens e *groundtruths* subamostrados – fornecidos como entrada para treinamento da rede.

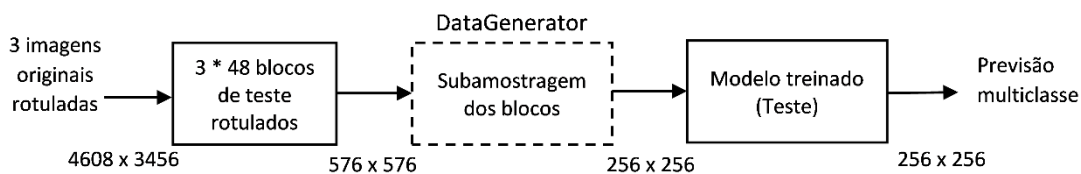


**Figura 5.23** – Esquema de geração de blocos de dados para treinamento das redes.

Durante o treinamento de cada modelo, os parâmetros (pesos) da rede são ajustados. Enquanto isso, as métricas de desempenho *loss* e IoU médio (*Mean Intersection over Union – mIoU*), descritas no [Apêndice C](#), são medidas no conjunto de treinamento e validação; armazenadas a cada época em um arquivo na pasta “*Checkpoints*”; e visualizadas em gráficos de desempenho no conjunto de treinamento e validação, usando a ferramenta Tensorboard [[TENSORBOARD](#)]. Os gráficos de desempenho dos experimentos de cada rede na configuração AT são apresentados na [Figura 5.31](#). O resultado de mIoU médio de validação e tempo de processamento de todos os experimentos são apresentados na [Tabela 5.6](#). No final do treinamento, os pesos treinados de cada rede, que minimizam o erro de previsão, são armazenados na pasta “*Weights*”, no arquivo nomeado “*model-base\_model.h5*”, onde *model* e *base\_model* são os nomes das redes treinadas.

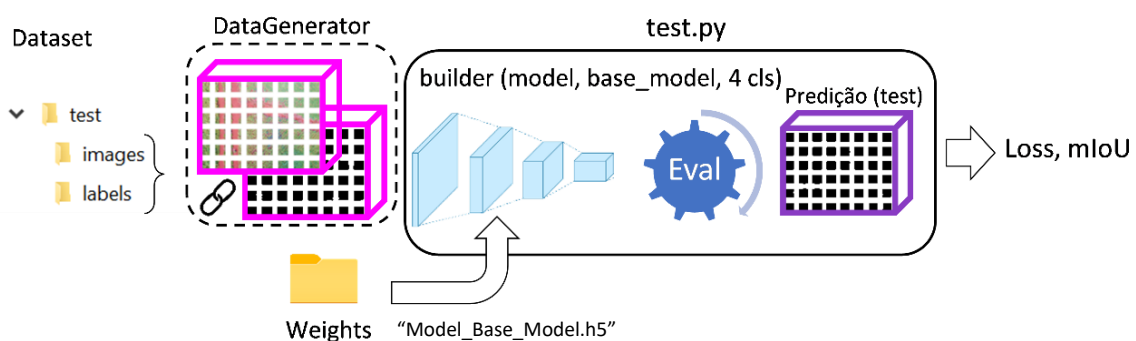
### 5.2.3.2 Teste com blocos de imagens RGB

Após o treinamento de cada rede de segmentação semântica, executamos o processo de teste dos modelos treinados, com objetivo de verificar a capacidade de generalização em um conjunto de dados de teste ainda não visto pela rede. O conjunto de teste é formado por  $8 \times 6$  blocos rotulados de 3 imagens aéreas originais RGB, opcionalmente subamostrados para  $256 \times 256$  pixels, sem aumento artificial de dados, como mostra a [Figura 5.24](#).



**Figura 5.24** – Esquema de geração de blocos de dados para teste das redes treinadas.

Para cada modelo de rede de segmentação semântica treinado, realizamos o processo de teste nas imagens RGB, como mostra a [Figura 5.25](#).



**Figura 5.25** – Processo de teste dos modelos de segmentação semântica a partir do conjunto de blocos de teste de três imagens RGB originais.

Para isso, executamos o código “*test.py*” com os seguintes parâmetros:

```
>> python test.py --model MODEL --base_model BASE_MODEL --num_classes NCLASSES --weights WEIGHTS --dataset DATASET ...
```



Os parâmetros completos do programa de teste são:

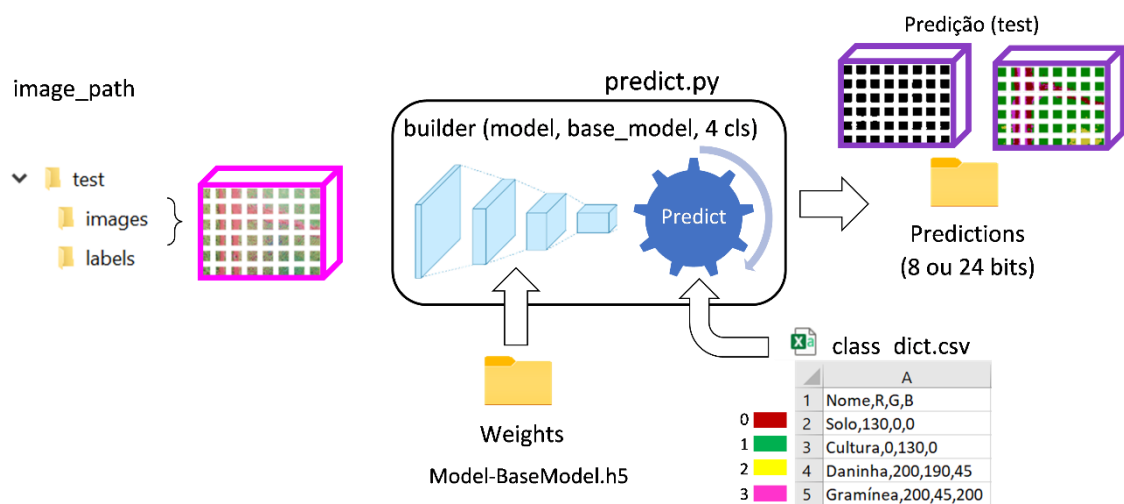
```

model          MODEL = {FCN-8s, UNet, SegNet, RefineNet, PSPNet, DenseASPP, DeepLabV3...}
base_model     BASE_MODEL = {VGG16, ResNet50, DenseNet121, Xception, Xception-DeepLab}
dataset       DATASET = Datasets/SugarcaneCTBE/ImagesRGB/576x576
num_classes   NCLASSES = 4
batch_size    BATCH_SIZE = 8
crop_height   CROP_HEIGHT = 256
crop_width    CROP_WIDTH = 256
weights       WEIGHTS = Networks/Weights/Model_BaseModel.h5
  
```

Os pesos da rede treinada são carregados do arquivo correspondente, definido pelo parâmetro *weights*. O módulo *DataGenerator* (com subamostragem e sem aumento de dados) gera a entrada a partir dos blocos rotulados armazenados na pasta “*test*”, localizada no caminho do parâmetro *dataset*. Em seguida, a rede gera previsões de 8 bits para cada bloco de imagem do conjunto de teste. As previsões são comparadas com os *groundtruths* de 8 bits correspondentes, gerando como saída medidas de desempenho total *loss* e mIoU. Os resultados de mIoU de teste de todos os experimentos são apresentados na [Tabela 5.6](#).

### 5.2.3.3 Predição com blocos de teste das imagens RGB

Em seguida, realizamos o processo de predição a partir do conjunto de blocos de teste das imagens RGB originais. Os arquivos dos blocos preditos de 8 e 24 bits são armazenados na pasta “*Predictions/predictions\_ImagesRGB\_Model\_BaseModel\_nbits*”, como mostrado na [Figura 5.26](#).



**Figura 5.26** – Processo de predição de 8 e 24 bits a partir do conjunto de blocos de teste de três imagens RGB originais.

Executamos o código “*predict.py*” no conjunto de teste, sem fornecer o *groundtruth* como entrada da rede de segmentação semântica:

```

>> python predict.py --model MODEL -- BASE_MODEL --num_classes NCLASSES --weights WEIGHTS
--image_path IMAGE_PATH --csv_file CSV_FILE --color_encode COLOR_ENCODE ...
  
```

Os parâmetros do programa de predição são:

```

model          MODEL = {FCN-8s, UNet, SegNet, RefineNet, PSPNet, DenseASPP, DeepLabV3...}
base_model     BASE_MODEL = {VGG16, ResNet50, DenseNet121, Xception, Xception-DeepLab}
image_path     IMAGE_PATH = Datasets/SugarcaneCTBE/ImagesRGB/576x576/test/images
num_classes    NCLASSES = 4
crop_height    CROP_HEIGHT = 256
crop_width     CROP_WIDTH = 256
weights        WEIGHTS = Networks/Weights/Model_BaseModel.h5
csv_file       CSV_FILE Datasets/SugarcaneCTBE/ImagesRGB/576x576/class_dict.csv
color_encode   COLOR_ENCODE = {True, False}

```

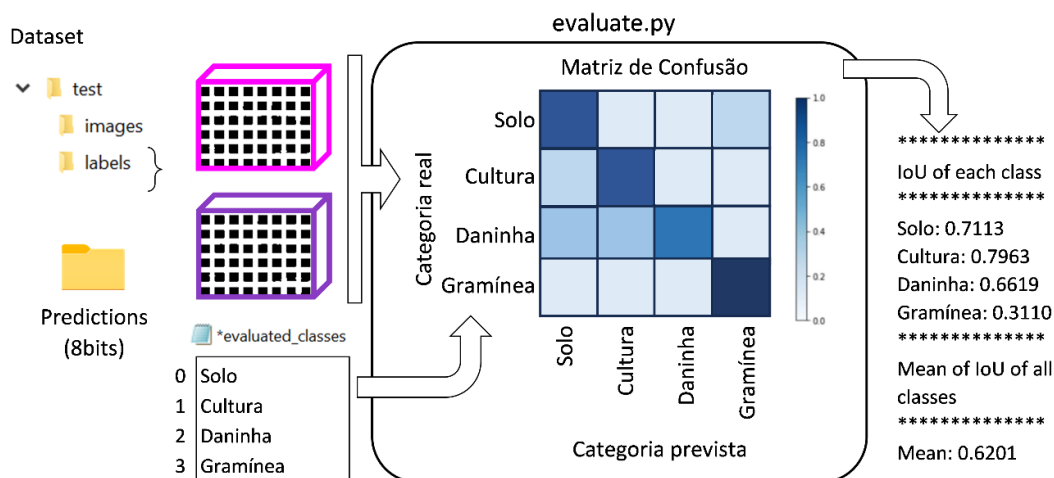
As entradas da rede são os blocos de teste (sem os *groundtruths* correspondentes), localizados na pasta definida pelo parâmetro *image\_path*, com subamostragem opcional. Os pesos da rede treinada são carregados do arquivo correspondente, definido pelo parâmetro *weights*. Para cada modelo de rede de segmentação, executamos o código “*predict.py*” duas vezes para gerar os blocos de predições de 8 ou 24 bits, armazenados na pasta “*Predictions/predictions\_ImagesRGB\_Model\_BaseModel\_nbits*”, de acordo com o parâmetro booleano *color\_encode*. Se *color\_encode* é *true*, a codificação de cores por classe é executada de acordo com a tabela de cores definida pelo arquivo “*class\_dict.csv*”, dado pelo parâmetro *csv-file*.

Os blocos preditos coloridos, subamostrados de  $256 \times 256$  *pixels*, são reagrupados na grade de  $8 \times 6$  blocos para recompor as predições de 24 bits de 3 imagens de teste, subamostradas para  $2.048 \times 1.536$  *pixels*. Por fim, uma avaliação qualitativa é realizada através da comparação visual das predições coloridas de teste com os respectivos *groundtruths* de 24 bits, como será apresentado na [Seção 5.3.2](#).

A seguir, uma avaliação quantitativa de desempenho mIoU para cada classe é realizada, conforme o procedimento descrito na próxima seção, por meio da comparação das predições de teste de 8 bits com os *groundtruths* de 8 bits correspondentes.

#### 5.2.3.4 Avaliação de desempenho com blocos de teste das imagens RGB

Para uma avaliação quantitativa mais detalhada de desempenho no conjunto de teste, realizamos um processo de avaliação por classe, conforme ilustra a [Figura 5.27](#).



**Figura 5.27** – Processo de avaliação dos modelos de segmentação semântica pela comparação das predições de teste de 8 bits com os blocos *groundtruths* correspondentes.

Executamos o código “*evaluate.py*” nos blocos preditos de 8 bits (gerados na pasta *predictions* na etapa anterior). Assim, as métricas de desempenho mIoU por classe e a média de mIoU de todas as classes são calculadas, comparando os blocos preditos com os *groundtruths* de 8 bits do conjunto de teste definido pelo parâmetro *dataset*.

```
>> python evaluate.py --dataset DATASET --predictions PREDICTIONS
```

Os parâmetros do programa de avaliação são:

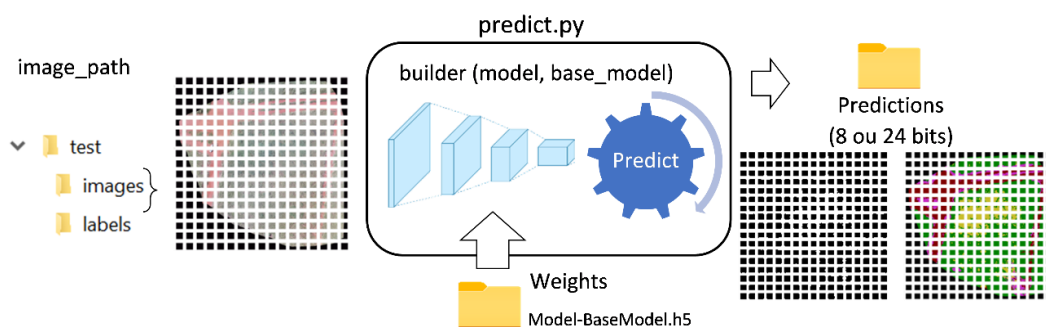
```
dataset          DATASET = Datasets/SugarcaneCTBE/ImagesRGB/576x576
crop_height      CROP_HEIGHT = 256
crop_width       CROP_WIDTH = 256
predictions      PREDICTIONS = Networks/Predictions/predictions_ImagesRGB_Model_BaseModel_8bits
```

Primeiro, a matriz de confusão multiclasse entre os *groundtruths* de 8 bits (na pasta “*test/labels*” definida no caminho do parâmetro *dataset*) e as previsões de 8 bits (na pasta definida pelo parâmetro *predictions*) é calculada por classe, para obter os valores de TP<sub>c</sub>, FP<sub>c</sub> e FN<sub>c</sub>, onde *c* corresponde a cada uma das classes definidas no arquivo “*evaluate\_classes.txt*” localizado na pasta do parâmetro *dataset*. A métrica mIoU de cada classe é calculada por TP<sub>c</sub> / (TP<sub>c</sub> + FP<sub>c</sub> + FN<sub>c</sub>), conforme descrito no [Apêndice C](#). Os resultados de desempenho mIoU por classe no conjunto de teste de imagens RGB serão apresentados na [Tabela 5.7](#).

### 5.2.3.5 Predição com blocos do mosaico RGB

Para obter uma avaliação completa do desempenho de cada rede sobre toda a área experimental de cultivo de cana-de-açúcar, os mesmos processos de predição e avaliação são executados nos blocos de teste do mosaico RGB.

Na [Figura 5.28](#), podemos ver o esquema de predição das redes de segmentação semântica a partir dos 17 × 17 blocos de 512 × 512 *pixels* do ortomosaico RGB (sem os *groundtruths*), opcionalmente<sup>13</sup> subamostrados para 256 × 256 *pixels* na entrada da rede. Cada modelo de rede usa os pesos já treinados no nosso conjunto de imagens aéreas RGB, para fazer a predição dos blocos do mosaico em 4 classes, visualizados em arquivos de 8 ou 24 bits, salvos na pasta “*Predictions/predictions\_MosaicRGB\_Model\_BaseModel\_nbits*”.



**Figura 5.28** – Processo de predição de 8 e 24 bits dos modelos de segmentação semântica a partir do conjunto de blocos de teste do mosaico RGB.

O código de execução “*predict.py*” para predição de 8 ou 24 bits é:

```
>> python predict.py --model MODEL --base_model BASE_MODEL --num_classes NCLASSES --weights
WEIGHTS --image_path IMAGE_PATH --csv_file --color_encode COLOR_ENCODE ...
```

<sup>13</sup> Outros testes descritos na Seção 5.3.1.3 foram realizados sem subamostragem.

Os parâmetros completos do programa de predição a partir dos blocos do mosaico RGB são:

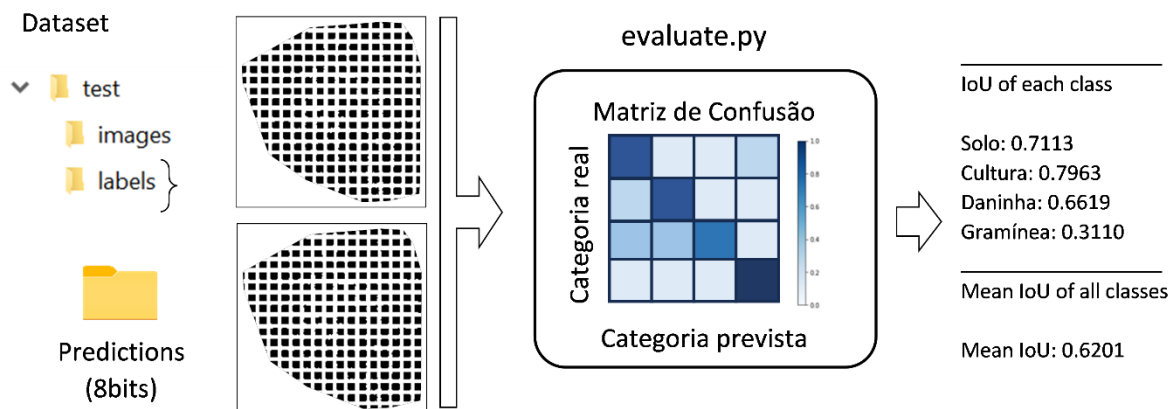
```

model          MODEL = {FCN-8s, UNet, SegNet, RefineNet, PSPNet, DenseASPP, DeepLabV3, ...}
base_model     BASE_MODEL = {VGG16, ResNet50, DenseNet121, Xception, Xception-DeepLab}
image_path     IMAGE_PATH = Datasets/SugarcaneCTBE/MosaicoRGB/512x512/test/images
num_classes    NUM_CLASSES = 4
crop_height    CROP_HEIGHT = 256
crop_width     CROP_WIDTH = 256
weights        WEIGHTS = Networks/Weights/Model_BaseModel.h5
csv_file       CSV_FILE = Datasets/SugarcaneCTBE/MosaicoRGB/512x512/class_dict.csv
color_encode   COLOR_ENCODE = {True, False}

```

### 5.2.3.6 Avaliação de desempenho com blocos do mosaico RGB

Após a predição de 8 bits, é realizada a avaliação de cada rede com blocos do mosaico completo (exceto os blocos inválidos nas bordas do mosaico), como mostra a [Figura 5.29](#). O código “*evaluate.py*” calcula as métricas de mIoU para cada classe, comparando os *groundtruths* de 8 bits (na pasta de teste definida pelo parâmetro *dataset*) e as previsões de 8 bits (na pasta definida pelo parâmetro *predictions*). Os resultados de mIoU por classe sobre o mosaico RGB serão apresentados na [Tabela 5.13](#).



**Figura 5.29** – Processo de avaliação dos modelos de segmentação semântica pela comparação das previsões de 8 bits dos blocos de teste do mosaico RGB e os *groundtruths* correspondentes.

O código de execução “*evaluate.py*” é:

```
>> python evaluate.py --dataset DATASET --predictions PREDICTIONS
```

Os parâmetros do programa de avaliação com blocos do mosaico RGB são:

```

dataset        DATASET = Datasets/SugarcaneCTBE/MosaicoRGB/512x512
crop_height    CROP_HEIGHT = 256
crop_width     CROP_WIDTH = 256
predictions    PREDICTIONS = Networks/Predictions/predictions_MosaicRGB_Model_BaseModel_8bits

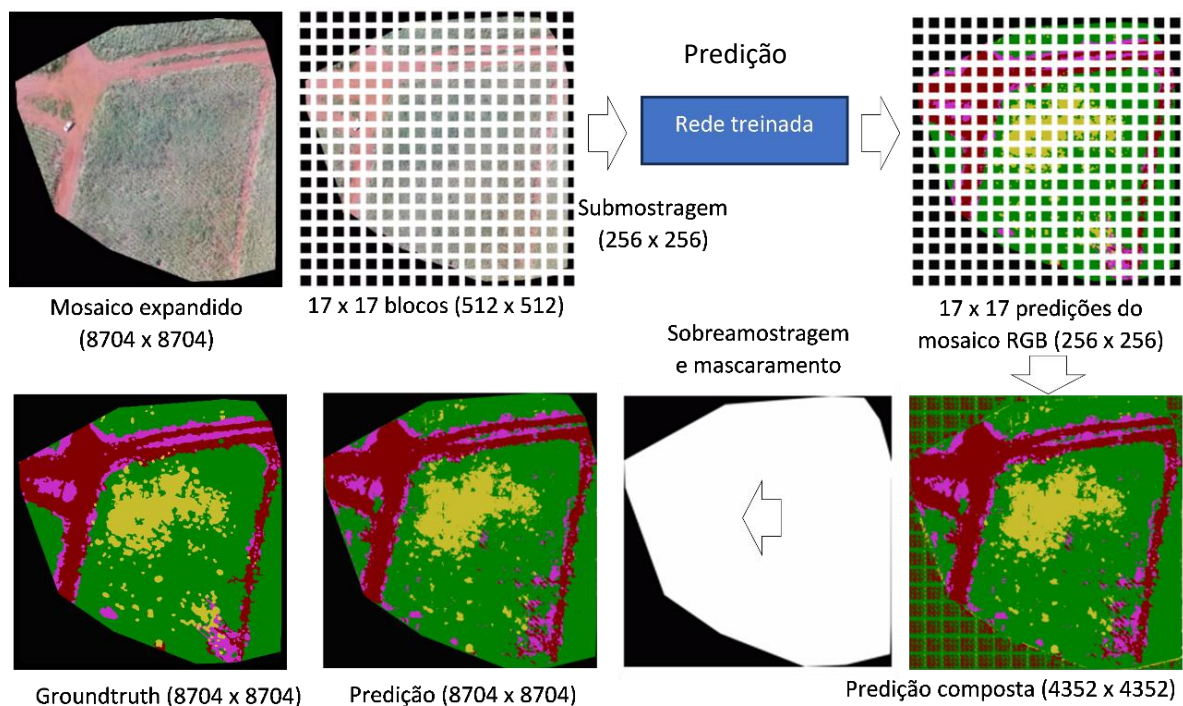
```

### 5.2.3.7 Composição dos blocos preditos do mosaico RGB

Após a predição dos blocos de 24 bits na [Seção 5.2.3.5](#), os blocos da grade devem ser recompostos para retornar ao tamanho de um mosaico completo. Na [Figura 5.30](#), os blocos preditos coloridos de  $256 \times 256$  pixels são reagrupados em uma grade de  $17 \times 17$  blocos para compor o mosaico predito de 24 bits, subamostrado para  $4.352 \times 4.352$  pixels. Em seguida, o mosaico predito completo pode ser sobreamostrado na resolução original de  $8.704 \times 8.704$  pixels.

Como o mosaico apresenta bordas pretas expandidas e irregulares de fundo, que não foram rotuladas e treinadas pela rede, os rótulos preditos nas bordas são imprevisíveis e inválidos, portanto, usamos uma máscara ([Figura 4.46b](#)) para zerar os pixels preditos de 24 bits nas bordas do mosaico composto. Pelo mesmo motivo, os blocos inválidos de 8 bits devem ser descartados do processo de avaliação de desempenho na [Figura 5.29](#), conforme já discutido da [Seção 5.1.4](#).

Para finalizar, o mosaico predito colorido completo de cada rede de segmentação é salvo em formato PNG de 24 bits para análise visual qualitativa na [Seção 5.3.4](#).



**Figura 5.30** – Composição dos  $17 \times 17$  blocos preditos coloridos de  $256 \times 256$  pixels, com mascaramento de borda para comparação com o groundtruth.

Seguindo os procedimentos descritos nesta [Seção 5.2](#), realizamos vários experimentos, com diferentes configurações de treinamento e teste, para avaliar as redes de segmentação semântica em nosso conjunto de dados. A seguir, analisamos os resultados de desempenho quantitativos e qualitativos obtidos.



## 5.3 Resultados das redes de segmentação semântica

Nesta seção, apresentamos uma análise quantitativa e qualitativa do desempenho de predição das nove redes de segmentação semântica selecionadas e treinadas usando nosso conjunto de dados de imagens aéreas de VANT de uma área de cultivo de cana-de-açúcar com presença de plantas daninhas. O [Apêndice B](#) ilustra os modelos de rede implementados e avaliados neste trabalho.

### 5.3.1 Resultados quantitativos com imagens aéreas RGB

Cada experimento listado na [Tabela 5.5](#) usa diferentes configurações de rede, tais como: modelo com inicialização de pesos aleatória identificado por A; modelo de segmentação com *backbone* pré-treinado no conjunto ImageNet de 1.000 classes sem reajuste dos pesos, referenciados por NT (*No Trainable*); ou *backbone* pré-treinado com todos os pesos reajustados no conjunto de dados de treinamento, referenciados por AT (*All Trainable*). Os resultados dos modelos treinados com aumento de dados foram referenciados por DA (*Data Augmentation*).

Os modelos de segmentação FCN, U-Net, SegNet, RefineNet e BiSeNet ([Figuras B.1 a B.5](#)) usam *backbone* com as cinco camadas semelhantes à CNN-base original. Desta forma, estas camadas são inicializadas com pesos aleatórios (A) ou por transferência de aprendizado de todas as camadas (NT e AT). Na configuração NT(5), os parâmetros de todas as cinco camadas da CNN não são treináveis. Na configuração AT(5), todos os parâmetros da CNN são reajustados no conjunto de dados destino.

As redes PSPNet, DeepLab-v3, DeepLab-v3+ e DenseASPP ([Figuras B.6 a B.12](#)) reutilizam apenas três ou quatro camadas iniciais da CNN original, redefinindo as camadas restantes do *backbone* com convolução *atrous* sem subamostragem. Assim, testamos as configurações AT(3) ou AT(4), dependendo da arquitetura da rede. Como não foram transferidas todas as camadas da CNN, a configuração NT(5) não foi executada; assim, as configurações NT(3) e NT(4) congelam as  $n$  primeiras camadas transferidas do *backbone*, e as camadas restantes são treináveis.

#### 5.3.1.1 Medidas de desempenho de treinamento e validação

Na [Tabela 5.5](#), são mostrados o número total de parâmetros, parâmetros treináveis e não treináveis, além do tempo de treinamento, de acordo com cada configuração.

Em relação ao número de parâmetros, podemos verificar que o modelo FCN-8s tem muito mais parâmetros do que outras redes, sendo 102.764.544 apenas na camada convolucionalizada FC6 com 4.096 filtros de convolução  $7 \times 7$  ([Apêndice B.1](#)). As redes codificadoras-decodificadoras U-Net e SegNet têm o dobro do número de parâmetros do *backbone* ([Apêndices B.2 e B.3](#)). A rede RefineNet tem mais parâmetros adicionais em relação ao *backbone* ([Apêndice B.4](#)). A rede BiSeNet acrescenta poucos parâmetros ao *backbone*, referentes ao caminho espacial, apresentando o menor tempo de treinamento ([Apêndice B.5](#)). A rede PSPNet tem um grande número de parâmetros adicionais, principalmente na convolução após a concatenação ([Apêndice B.6](#)). As redes DeepLab-v3 e v3+ acrescentam um número razoável de parâmetros na pirâmide ASPP ([Apêndice B.7 e B.8](#)). A rede DenseASPP ([Apêndice B.9](#)) tem o menor número de parâmetros usando *backbone* DenseNet-121, mas não o menor tempo de treinamento. Na configuração AT

usando *backbone* ResNet-50, o número de parâmetros é maior, embora não tenha muito acréscimo de parâmetros no módulo denso.

Todos os experimentos na [Tabela 5.5](#) foram realizados em uma GPU de menor velocidade NVIDIA® GeForce GTX TITAN X com 12GB de memória RAM (CUDA 11.4) (marcada em asterisco) ou GPU NVIDIA™ RTX A5000 com 24 GB de memória RAM (CUDA 11.6) de maior velocidade.

**Tabela 5.5** – Diferentes configurações das redes de segmentação semântica baseadas em CNN, número de parâmetros e tempo de treinamento.

Modelo de Segmentação	CNN-base (Parâmetros)	Parâmetros total	Parâmetros treináveis	Parâmetros não treináveis	Tempo de treinamento	Configuração de rede
FCN-8s	VGG-16	134.284.632	134.284.632	0	*4 h 22 min	A
		134.284.632	134.284.632	0	*4 h 17 min	A/DA
		134.284.632	119.569.944	14.714.688	*2 h 58 min	NT(5)/DA
		134.284.632	134.284.632	0	*4 h 17 min	AT(5)/DA
U-Net	VGG-16 14.714.688 sem FC	25.866.388	25.860.620	5.768	5 h 24 min	A
		25.866.388	25.860.620	5.768	5 h 15 min	A/DA
		25.866.388	11.145.932	14.720.456	3 h 03 min	NT(5)/DA
		25.866.388	25.860.620	5.768	5 h 29 min	AT(5)/DA
SegNet	VGG-16 14.714.688 sem FC	29.442.260	29.434.828	7.432	6 h 44 min	A
		29.442.260	29.434.828	7.432	6 h 48 min	A/DA
		29.442.260	14.720.140	14.722.120	4 h 34 min	NT(5)/DA
		29.442.260	29.434.828	7.432	*7 h 30 min	AT(5)/DA
RefineNet	ResNet-50 23.587.712	58.797.444	58.744.324	53.120	5 h 01 min	A
		58.797.444	58.744.324	53.120	*5 h 42 min	A/DA
		58.797.444	35.209.732	23.587.712	3 h 21 min	NT(5)/DA
	VGG-16 sem FC	49.301.828	34.587.140	14.714.688	4h 18 min	NT(5)/DA
		58.797.444	58.744.324	53.120	4 h 54 min	AT(5)/DA
BiSeNet	Xception 20.861.480	26.081.488	26.020.504	60.984	3 h 15 min	A
		26.081.488	26.020.504	60.984	*3 h 42 min	A/DA
		26.081.488	5.213.552	20.867.936	1 h 25 min	NT(5)/DA
		26.081.488	26.020.504	60.984	2 h 59 min	AT(5)/DA
PSPNet	ResNet-50 23.587.712	46.671.252	46.613.004	58.248	8 h 57 min	A
		46.671.252	46.613.004	58.248	*9 h 47 min	A/DA
		46.671.252	45.163.020	1.508.232	8 h 53 min	NT(3)/DA
		46.671.252	46.613.004	58.248	8 h 48 min	AT(3)/DA
DeepLab-v3	ResNet-50 23.587.712	39,123,332	39,069,700	53,632	26 h 38 min	A
		39,123,332	39,069,700	53,632	26 h 30 min	A/DA
		39,123,332	30,511,108	8,612,224	*27 h 35 min	NT(4)/DA
		39,123,332	39,069,700	53,632	25 h 40 min	AT(4)/DA
DeepLab-v3+	Xception-DeepLab	54.403.732	54.305.540	98.192	27 h 37min	A
		54.403.732	54.305.540	98.192	*28 h 32min	A/DA
	ResNet-50	40.428.660	40.373.908	54.752	26 h 50 min	AT(4)/DA
DenseASPP	DenseNet-121	9.284.484	9.186.756	97.728	* 6 h 48 min	A
		9.284.484	9.186.756	97.728	* 6 h 56 min	A/DA
	ResNet-50 23.587.712	27.169.988	25.642.564	1.527.424	8 h 30 min	NT(3)/DA
		27.169,988	27.092.548	77.440	8 h 43 min	AT(3)/DA

A – Inicialização aleatória, DA – Data Augmentation, NT – No Trainable, AT – All Trainable. \*GPU de menor velocidade

A [Tabela 5.5](#) mostra que a configuração selecionada influencia no tempo de treinamento, de acordo com a quantidade de parâmetros treináveis; por exemplo, a configuração NT tem o menor tempo em relação às outras configurações. No entanto, o tempo de treinamento depende não apenas do número de parâmetros treináveis, como também da quantidade de operações computacionais realizadas. Sendo assim, modelos com aproximadamente o mesmo número de parâmetros treináveis podem ter tempos de treinamento diferentes, como BiSeNet e U-Net.

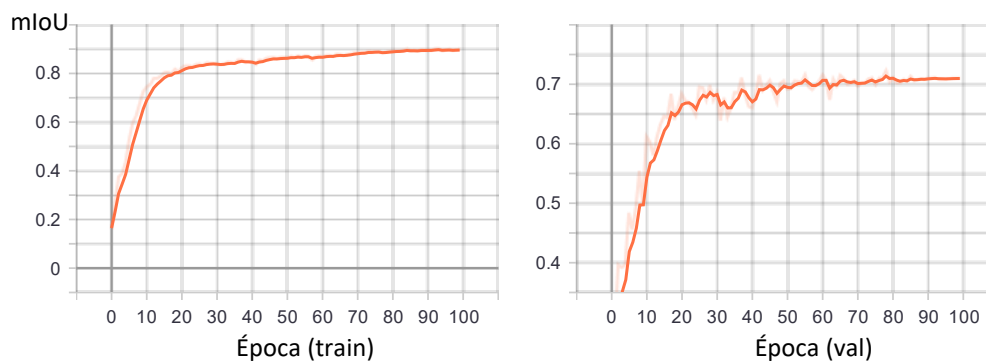
Durante o procedimento de treinamento/validação das redes de segmentação, as medidas de desempenho (mIoU) de treinamento e validação foram monitoradas, conforme procedimento descrito na [Seção 5.2.3.1](#). O valor mIoU é calculado pelo valor médio do IoU da predição dos blocos das três imagens RGB que compõem o conjunto de validação ou teste. Quanto maior o mIoU em porcentagem, melhor o desempenho da predição.

Usamos a ferramenta *TensorBoard* [[ABADI et al. 2015](#)], que permite visualizar as métricas de desempenho durante o treinamento. Na [Figura 5.31](#) podemos ver o gráfico de desempenho (mIoU  $\times$  época) de treinamento e validação de cada modelo de segmentação na configuração AT/DA. A maioria das redes levaram aproximadamente 30 épocas para começar a convergir, de forma gradual, enquanto a rede BiSeNet ([Figura 5.31e](#)) teve uma melhora de convergência a partir da metade do treinamento. Porém, todas as redes convergiram satisfatoriamente após 100 épocas de treinamento.

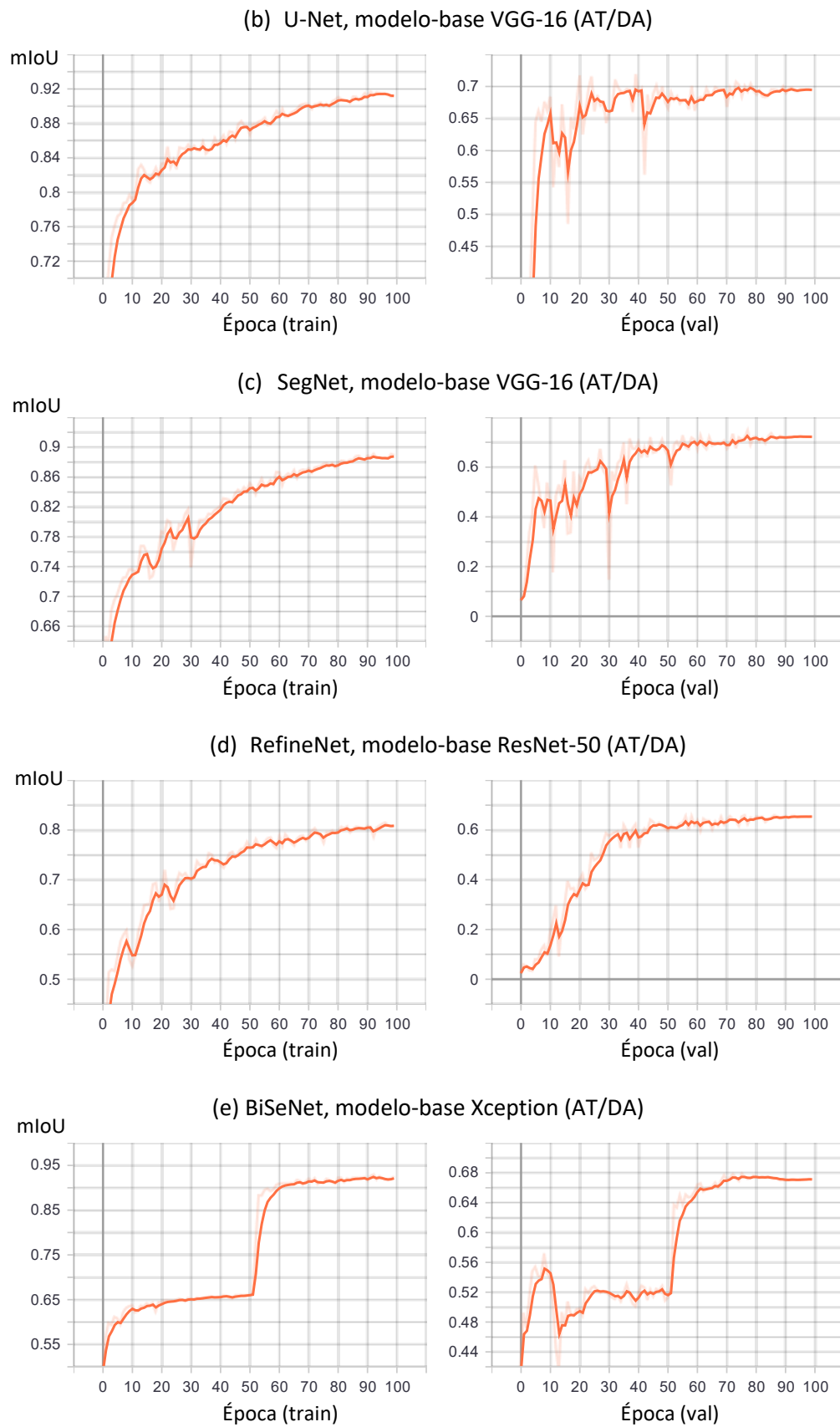
Observamos que, em geral, embora o desempenho continue aumentando durante o treinamento, com IoU médio de treinamento superior a 80% em todas as redes, o mIoU de validação não ultrapassa 72% ([Figura 5.31c](#)). Isso acontece devido à quantidade limitada de dados para treinamento da rede, mesmo com transferência de aprendizado e aumento artificial de dados. O passo opcional de subamostragem da entrada de  $512 \times 512$  para  $256 \times 256$  pixels (utilizada para redução do tempo de treinamento) também reduz ligeiramente o mIoU de treinamento e validação, como veremos na [Figura 5.40a e d](#).

O treinamento foi executado por 100 épocas para evitar sobreajuste excessivo e problemas de generalização no conjunto de teste; em outras palavras, isso evita que a rede tenha um aumento de desempenho no treinamento, mas com aumento da degradação do desempenho de teste, em dados nunca vistos pela rede, conforme discutido na [Seção 2.1.2](#).

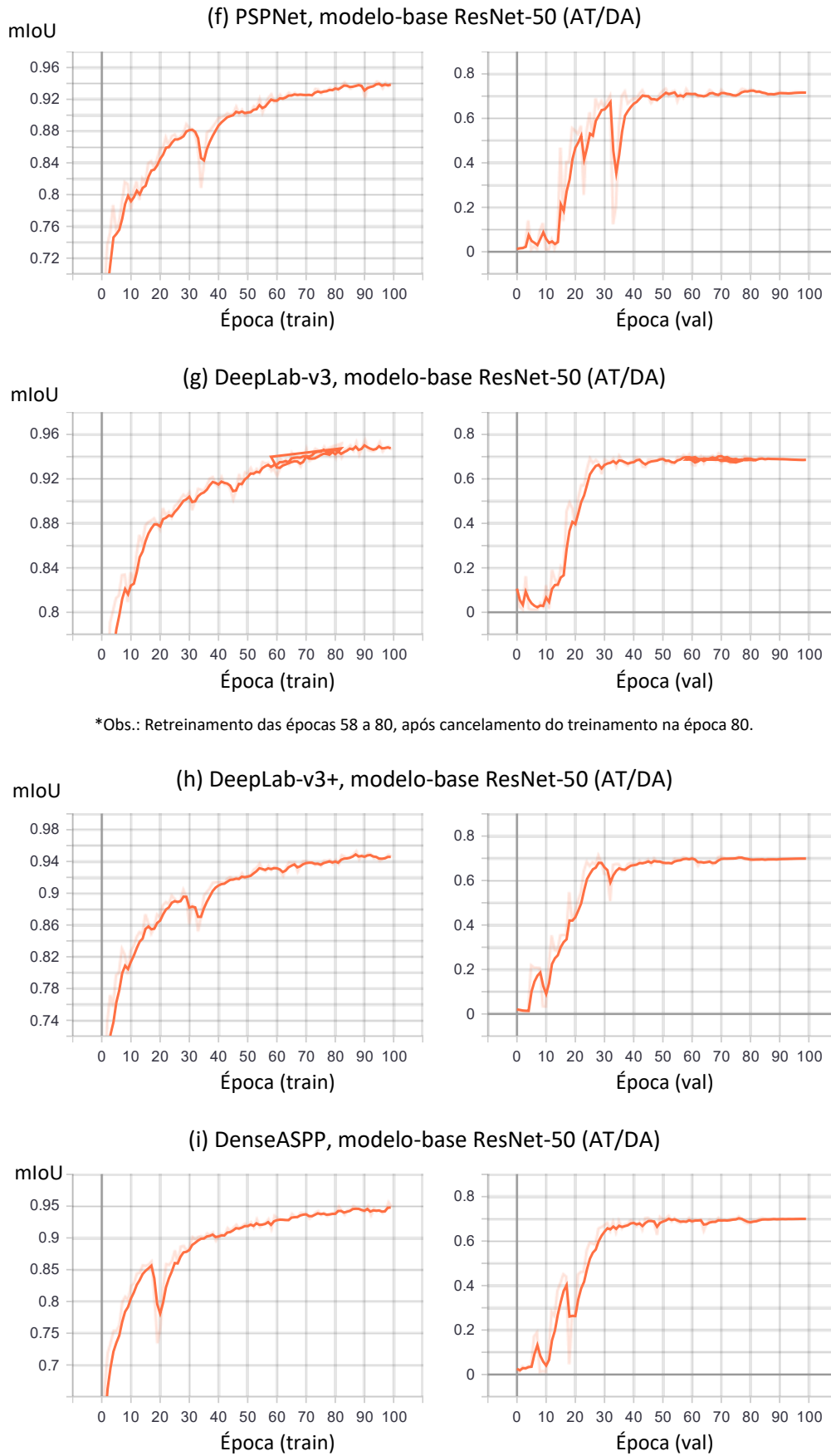
(a) FCN-8s, modelo-base VGG-16 (AT/DA)



**Figura 5.31** – *Tensorboard*: desempenho mIoU de treinamento e validação (continuação).



**Figura 5.31** – Tensorboard: desempenho mIoU de treinamento e validação (continuação).



**Figura 5.31** – Tensorboard: desempenho mIoU de treinamento e validação.

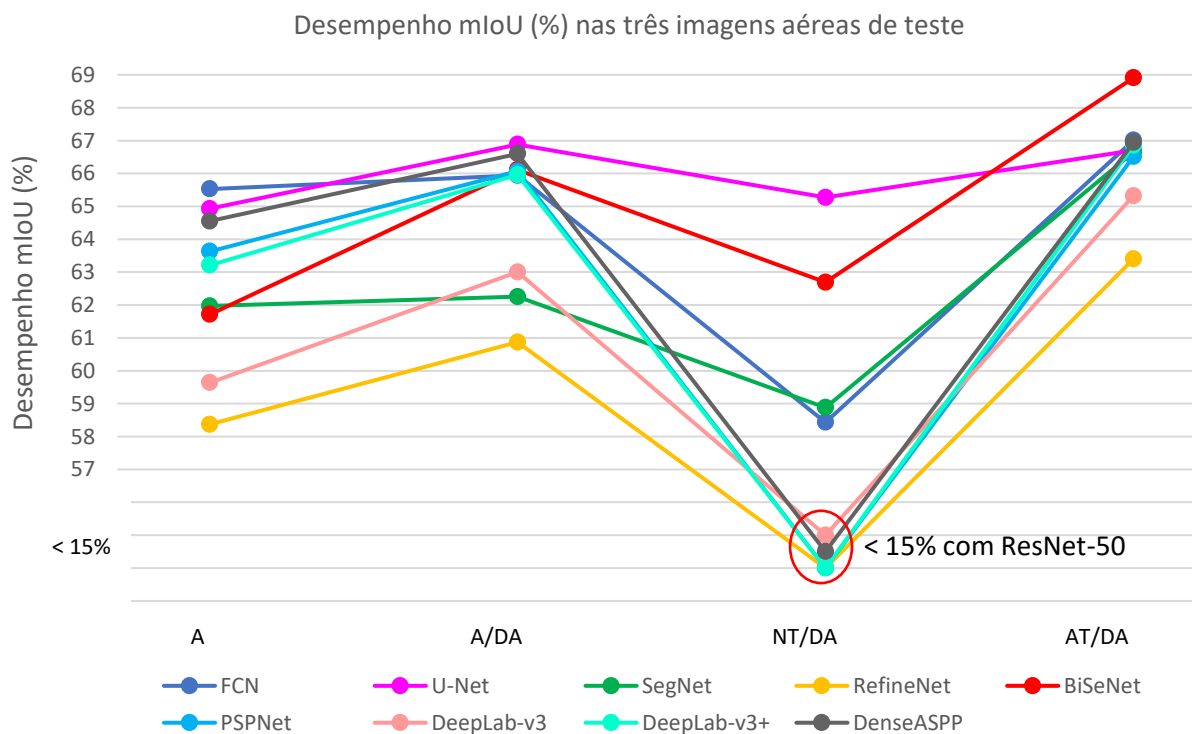


### 5.3.1.2 Medidas de desempenho de teste

Após o treinamento, foram realizadas as avaliações no conjunto de blocos de teste das três imagens aéreas originais RGB, conforme discutido na [Seção 5.2.3.2](#). Os resultados quantitativos de desempenho mIoU de validação e mIoU de teste (utilizando os pesos treinados na época 100) são apresentados na [Tabela 5.6](#). Em geral, o mIoU de teste foi menor que mIoU de validação, e a configuração AT/DA de cada modelo apresentou melhores resultados, destacados em negrito.

Para facilitar a visualização, o gráfico na [Figura 5.32](#) mostra uma comparação de desempenho de mIoU de teste das diferentes configurações de inicialização de pesos de cada modelo. Devido ao pequeno tamanho do conjunto de dados de treinamento, o aumento artificial de dados na configuração A/DA melhorou os resultados das redes de segmentação em relação à configuração A, com inicialização de pesos aleatória sem aumento de dados; por este motivo, todos os treinamentos seguintes com transferência de aprendizado foram realizados com aumento de dados (NT/DA ou AT/DA).

Podemos observar no gráfico da [Figura 5.32](#) que, na configuração NT, apesar da maior velocidade de treinamento e menor possibilidade de sobreajuste devido ao menor número de parâmetros treináveis (isto é, todas ou quase todas as camadas do *backbone* congeladas), usar a CNN pré-treinada como extrator de atributos apresenta o pior desempenho devido às diferenças entre o conjunto ImageNet e as imagens aéreas RGB. Além disso, as redes que utilizaram a rede ResNet-50 como *backbone* tiveram um desempenho de mIoU de teste inferior a 15%, como destacado em vermelho na [Tabela 5.6](#). Por exemplo, a rede RefineNet na configuração NT(5)/DA teve desempenho de 2,52% com *backbone* ResNet-50 e 62,02% com VGG-16.



**Figura 5.32** – Comparação de desempenho de teste das diferentes configurações de rede usando aumento de dados (DA), inicialização aleatória de pesos (A), transferência sem ajuste dos pesos (NT) ou transferência com ajuste de todas os pesos (AT).

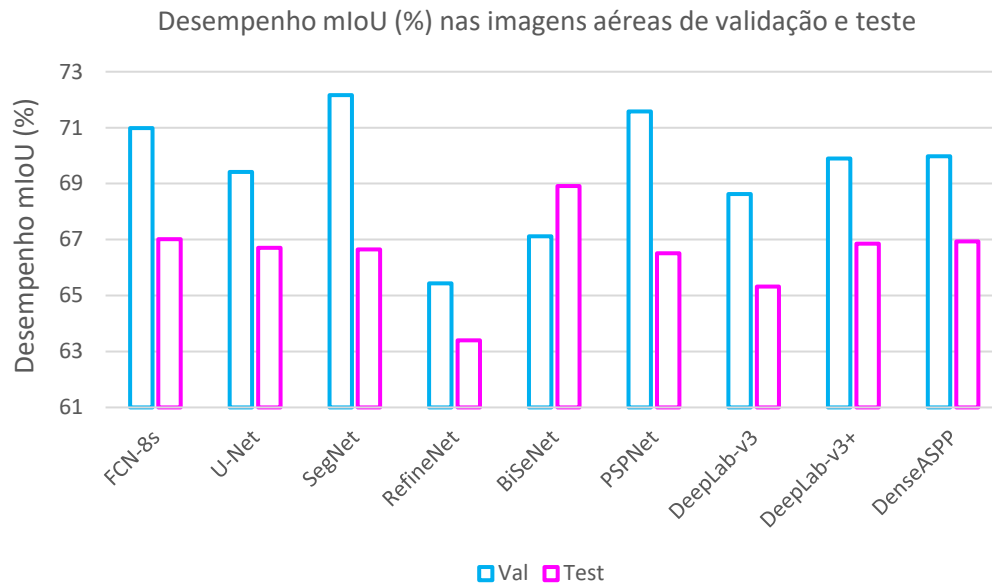
Por outro lado, as redes com pré-treinamento e ajuste de todos os pesos da CNN na configuração AT/DA obtiveram melhor desempenho, destacados em negrito na [Tabela 5.6](#). Isso mostra que atributos semânticos podem ser transferidos entre conjuntos de dados de base e destino, mesmo com um reajuste no conjunto de dados alvo. Desta forma, foi confirmada a expectativa de que a transferência de aprendizado com reajuste de todos os pesos seria melhor do que a inicialização aleatória, conforme discutido na [Seção 2.5.2](#).

**Tabela 5.6** – Desempenho IoU médio das redes de segmentação semântica no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada.

Modelo de Segmentação	CNN-base	mIoU (%)		Configuração de rede
		Val	Test	
FCN-8s	VGG-16	64,85	65,53	A
		64,45	65,94	A/DA
		56,54	58,43	NT(5)/DA
		<b>70,99</b>	<b>67,01</b>	AT(5)/DA
U-Net	VGG-16	65,06	64,93	A
		<b>69,48</b>	<b>66,88</b>	A/DA
		67,61	65,27	NT(5)/DA
		69,42	66,70	AT(5)/DA
SegNet	VGG-16	63,41	61,97	A
		62,90	62,25	A/DA
		58,69	58,88	NT(5)/DA
		<b>72,16</b>	<b>66,65</b>	AT(5)/DA
RefineNet	ResNet-50	55,90	58,37	A
		53,40	60,87	A/DA
		<b>1,32</b>	<b>2,52</b>	NT(5)/DA
	VGG-16	62,09	62,02	NT(5)/DA
		<b>65,43</b>	<b>63,40</b>	AT(5)/DA
BiSeNet	Xception	58,55	61,71	A
		65,67	66,11	A/DA
		56,86	62,69	NT(5)/DA
		<b>67,12</b>	<b>68,91</b>	AT(5)/DA
PSPNet	ResNet-50	63,00	63,63	A
		64,64	66,07	A/DA
		<b>1,53</b>	<b>2,79</b>	NT(3)/DA
		<b>71,58</b>	<b>66,51</b>	AT(3)/DA
DeepLab-v3	ResNet-50	54,89	59,54	A
		61,98	63,00	A/DA
		<b>12,83</b>	<b>12,53</b>	NT(4)/DA
		<b>68,63</b>	<b>65,32</b>	AT(4)/DA
DeepLab-v3+	Xception-DeepLab	60,32	63,21	A
		64,99	65,96	A/DA
	ResNet-50	<b>69,90</b>	<b>66,85</b>	AT(4)/DA
DenseASPP	DenseNet-121	62,59	64,55	A
		66,82	66,60	A/DA
	ResNet-50	<b>4,45</b>	<b>5,69</b>	NT(3)/DA
		<b>69,98</b>	<b>66,94</b>	AT(3)/DA

A – Inicialização aleatória, DA – Data Augmentation, NT – No Trainable, AT – All Trainable

O gráfico da [Figura 5.33](#) apresenta os valores de desempenho mIoU da melhor configuração de cada modelo (AT/DA) na tarefa de segmentação semântica, usando as imagens aéreas de validação e teste. Todos os modelos apresentaram desempenhos com valores de IoU médio de teste entre 65,32% e 68,91%, exceto RefineNet com menor valor de IoU médio de teste de 63,40%. Dentre todos os modelos, a rede BiSeNet obteve o melhor desempenho de IoU médio de teste de 68,91%, com o menor tempo de treinamento.

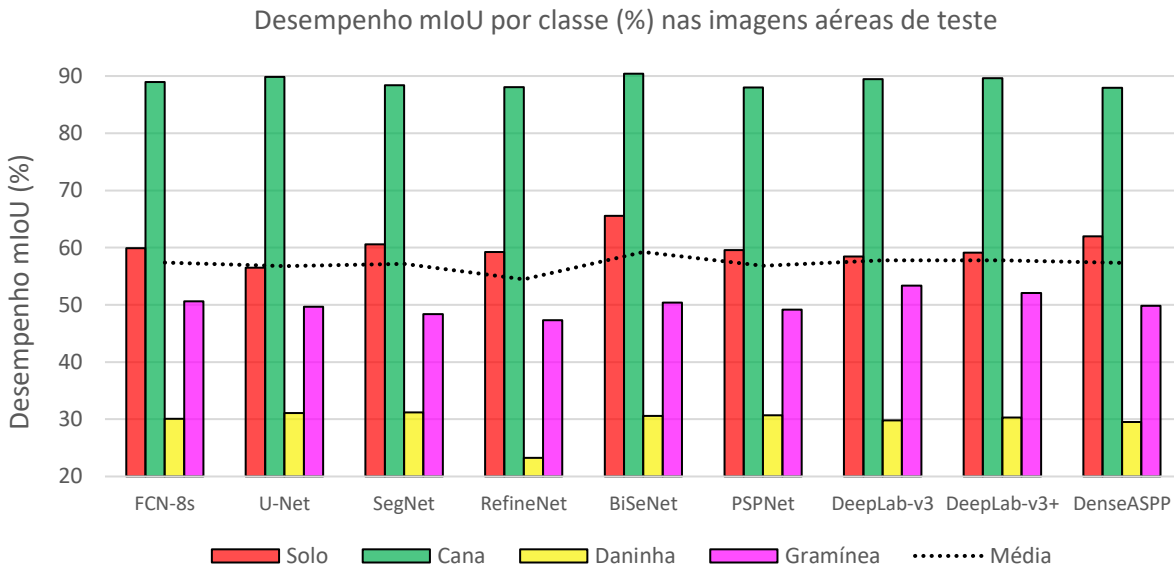


**Figura 5.33** – Desempenho mIoU da melhor configuração (AT/DA) de cada modelo no conjunto de imagens aéreas de validação e teste.

Os resultados quantitativos de IoU médio por classe, obtidos no procedimento de avaliação no conjunto de imagens aéreas de teste, discutido na [Seção 5.2.3.4](#), são apresentados na [Tabela 5.7](#) e mostrados no gráfico da [Figura 5.34](#). Embora todas as redes obtiveram resultados parecidos, a rede BiSeNet obteve o melhor desempenho médio global, mantendo uma precisão competitiva em todas as classes, sendo a média por classe e a média global maiores quando comparadas com métodos mais complexos, como DeepLab.

**Tabela 5.7** – Desempenho de mIoU por classe e mIoU médio das redes de segmentação semântica no conjunto de teste composto por blocos de três imagens aéreas RGB.

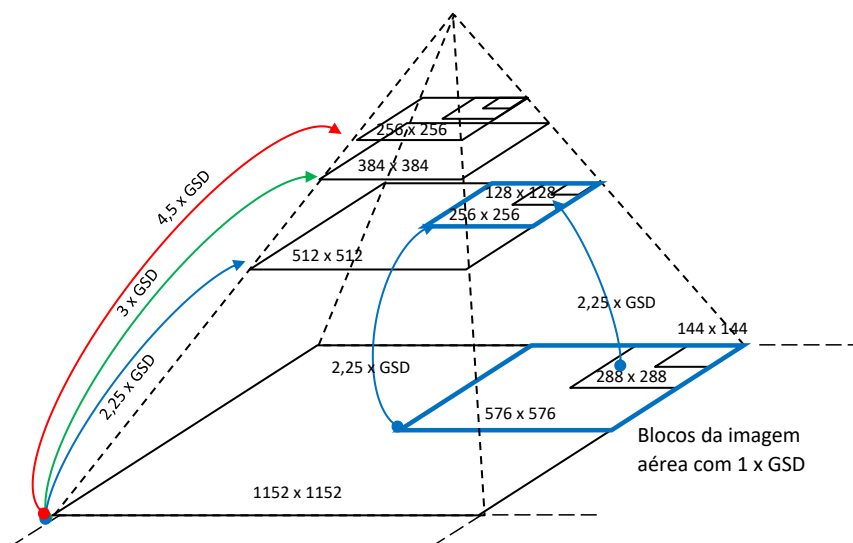
Modelo de Segmentação	CNN-base	Solo (%)	Cana (%)	Daninha (%)	Gramínea (%)	mIoU médio	Configuração de rede
FCN-8s	VGG-16	59,93	88,93	30,08	50,61	57,39	AT(5)/DA
U-Net	VGG-16	56,50	89,83	31,10	49,69	56,78	AT(5)/DA
SegNet	VGG-16	60,59	88,38	<b>31,23</b>	48,39	57,15	AT(5)/DA
RefineNet	ResNet-50	59,26	88,06	23,23	47,34	54,47	AT(5)/DA
BiSeNet	Xception	<b>65,57</b>	<b>90,39</b>	30,57	50,42	<b>59,24</b>	AT(5)/DA
PSPNet	ResNet-50	59,55	88,00	30,71	49,16	56,86	AT(3)/DA
DeepLab-v3	ResNet-50	58,43	89,47	29,81	<b>53,39</b>	57,77	AT(4)/DA
DeepLab-v3+	ResNet-50	59,10	89,60	30,28	52,10	57,77	AT(4)/DA
DenseASPP	ResNet-50	61,96	87,94	29,54	49,85	57,33	AT(3)/DA



**Figura 5.34** – Desempenho *mIoU* por classe e *mIoU* médio (linha tracejada) da melhor configuração (AT/DA) de cada modelo no conjunto de imagens aéreas de teste.

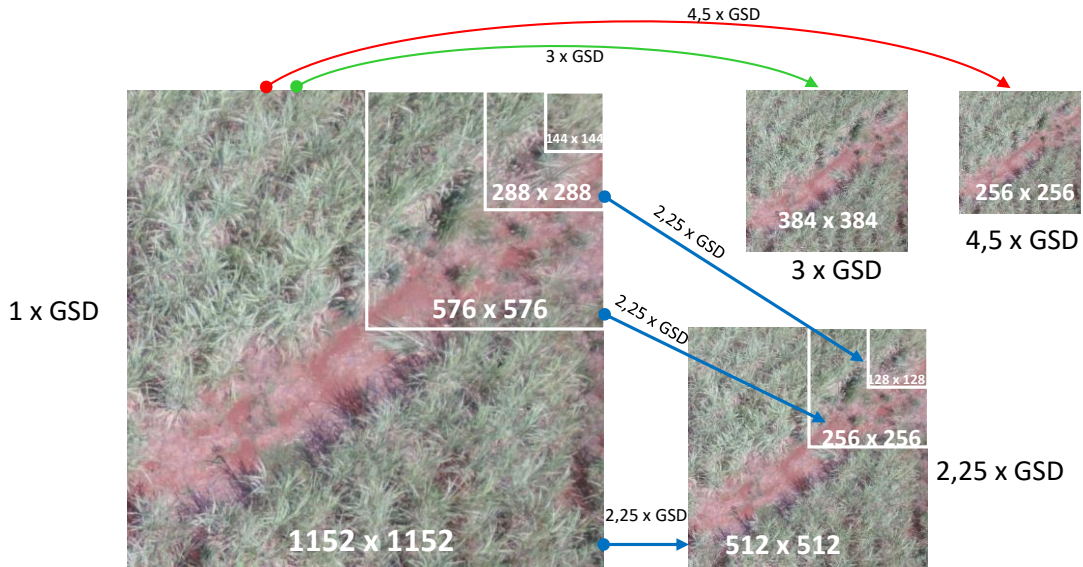
### 5.3.1.3 Testes adicionais em diferentes resoluções

Todos os testes até agora foram realizados com blocos de entrada na mesma resolução dos blocos de treinamento, isto é,  $576 \times 576$  pixels subamostrados para  $256 \times 256$  pixels. Para testar a capacidade de generalização das redes em diferentes resoluções, avaliamos o desempenho usando como entrada blocos de teste com diferentes tamanhos de recorte e GSD. Exemplificando, como as imagens aéreas RGB têm um GSD médio de  $1,16 \text{ cm/pixel}$ , as redes treinadas e testadas com recortes de tamanho 576 subamostrados para 256 (taxa de subamostragem  $576/256 = 2,25$ ), na verdade, extraem atributos de imagens com GSD de aproximadamente  $2,25 \times 1,16 \text{ cm/pixel}$ . Embora recortes da imagem aérea de diferentes tamanhos contenham informação de áreas de diferentes tamanhos no solo, os testes das redes com blocos de  $1.152$  subamostrados para  $512$  ou blocos  $288$  subamostrados para  $128$  usam o mesmo GSD com que a rede foi treinada (como mostram as setas azuis na Figura 5.35); por isso, apresentam desempenhos similares na Tabela 5.8.



**Figura 5.35** – Relação dos recortes subamostrados para obter diferentes GSDs.

A [Figura 5.36](#) mostra exemplos de blocos de teste, onde os recortes originais de diferentes tamanhos na imagem de teste têm resolução espacial de 1 x GSD, mas são subamostrados em várias taxas para obter diferentes GSDs.



**Figura 5.36** – Blocos de teste com diferentes tamanhos e GSDs.

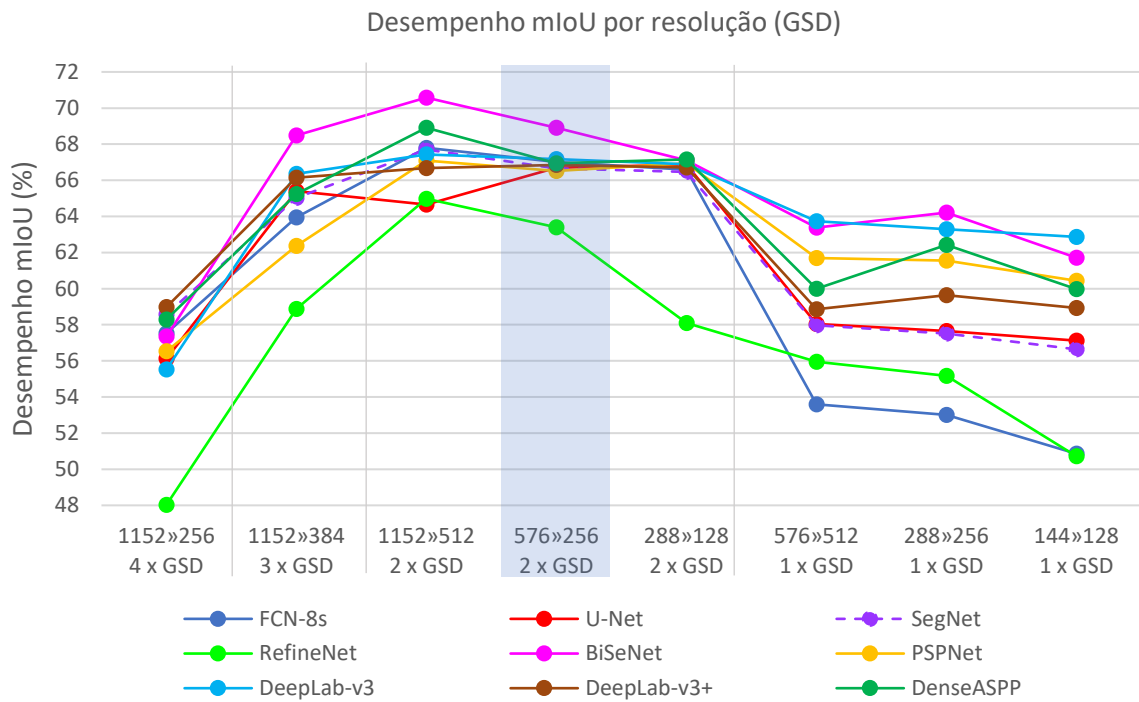
A [Tabela 5.8](#) mostra os resultados de desempenho dos modelos na configuração AT/DA, treinadas com blocos de tamanho 576 × 576 subamostrados para 256 × 256 (representado por 576 » 256) com resolução de 2,25 x GSD e testados em diversos tamanhos de bloco e GSD. Na primeira coluna da tabela, recortes de 1.152 × 1.152 nas três imagens aéreas RGB de teste de 4.608 × 3.456 pixels, geram uma grade de 4 × 3 blocos em cada imagem (com o mesmo GSD da imagem original), ou seja, temos um conjunto de 36 recortes (3 x 4 x 3) com resolução espacial de 1 x GSD. Subamostrando este conjunto de recortes de 1.152 × 1.152 para 256 × 256 pixels (1152 » 256), temos um conjunto de 36 blocos de teste com tamanho de 256 × 256 na entrada da rede com resolução de 4,5 x GSD. A coluna em azul mostra os resultados de desempenho de teste, usando blocos de teste com mesmo tamanho de recorte e GSD dos blocos com que a rede foi treinada.

**Tabela 5.8** – Desempenho de teste das redes (AT/DA) com diferentes GSDs de entrada.

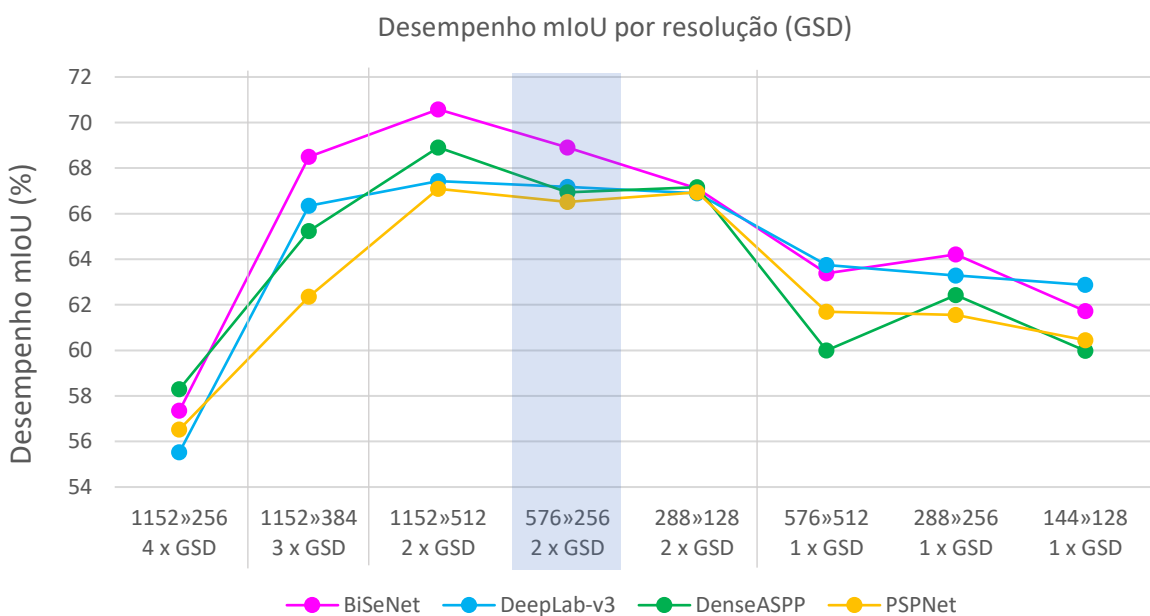
Modelo de segmentação	mIoU (%) de 3 imagens RGB de teste (4.608 x 3.456 pixels)							
	1152 » 256 3x(4x3) blocos 4 x GSD	1152 » 384 36 blocos 3 x GSD	1152 » 512 36 blocos 2 x GSD	576 » 256 144 blocos 2 x GSD	288 » 128 576 blocos 2 x GSD	576 » 512 144 blocos 1 x GSD	288 » 256 576 blocos 1 x GSD	144 » 128 2304 blocos 1 x GSD
FCN-8s	57,53	63,94	67,80	67,01	66,58	53,60	53,00	50,86
U-Net	56,12	65,40	64,65	66,70	66,87	58,04	57,66	57,12
SegNet	58,57	65,03	67,70	66,65	66,47	57,97	57,51	56,64
RefineNet	48,03	58,87	64,98	63,40	58,09	55,96	55,17	50,73
BiSeNet	57,35	68,49	70,58	68,91	67,11	63,38	64,21	61,72
PSPNet	56,53	62,36	67,09	66,51	66,94	61,69	61,55	60,44
DeepLab-v3	55,53	66,35	67,43	67,18	66,89	63,74	63,29	62,87
DeepLab- v3+	58,98	66,14	66,68	66,85	66,71	58,86	59,63	58,93
DenseASPP	58,30	65,24	68,91	66,94	67,16	60,00	62,43	59,98



Na [Figura 5.37](#), podemos ver o gráfico correspondente à [Tabela 5.8](#), com desempenho de teste de todas as redes na configuração AT/DA. Os modelos BiSeNet, DeepLab-v3, DenseASPP e PSPNet apresentam melhores resultados, uma vez que geram atributos multiescala, conforme discutido na [Seção 2.4.12](#). Desta forma, em comparação com outros modelos de conexão de salto ou codificador-decodificador, tais como FCN, RefineNet, U-Net e SegNet, eles conseguem obter melhores desempenhos de predição usando imagens de entrada com GSDs diferentes do originalmente treinado pela rede. A [Figura 5.38](#) mostra uma comparação destas quatro redes separadas, evidenciando as redes DeepLab-v3 e BiSeNet com resultados superiores.



**Figura 5.37** – Comparação de desempenho das redes com diferentes resoluções de entrada.

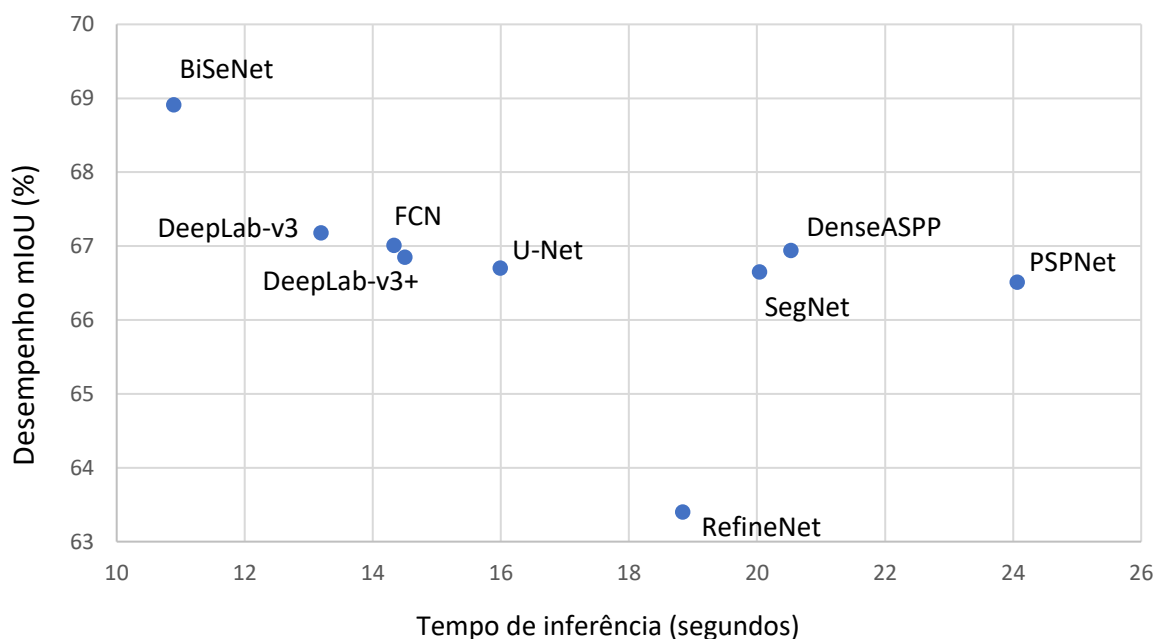


**Figura 5.38** – Comparação das quatro redes de melhor desempenho em todas as resoluções.

A [Tabela 5.9](#) mostra o tempo de execução de teste, que depende não apenas da arquitetura e do número de parâmetros de cada modelo, como também do tamanho dos blocos de entrada e do número de blocos no conjunto de teste. No gráfico da [Figura 5.39](#), gerado a partir dos valores de (desempenho  $\times$  tempo) da coluna azul da [Tabela 5.8](#) (eixo vertical) versus [Tabela 5.9](#) (eixo horizontal), podemos ver que as redes BiSeNet e DeepLab apresentam melhor desempenho e menor tempo de inferência no conjunto de 144 blocos de 576 subamostrados para 256 das três imagens aéreas de teste. Esses métodos se enquadram na área superior esquerda do gráfico com alta precisão e alta velocidade. As redes localizadas na parte inferior direita do gráfico têm menor velocidade e menor desempenho. A rede FCN-8s, apesar de ter boa velocidade de inferência, apresenta um tempo elevado para carregamento dos pesos treinados (não incluído no tempo de execução).

**Tabela 5.9** – Tempo de execução dos testes das redes com diferentes resoluções de entrada.

Modelo	Tempo de teste (em segundos) de 3 imagens RGB (4.608 x 3.456 pixels)							
	1152 » 256 36 blocos 4 x GSD	1152 » 384 36 blocos 3 x GSD	1152 » 512 36 blocos 2 x GSD	576 » 256 144 blocos 2 x GSD	288 » 128 576 blocos 2 x GSD	576 » 512 144 blocos 1 x GSD	288 » 256 576 blocos 1 x GSD	144 » 128 2304 blocos 1 x GSD
FCN-8s	4,14	11,68	8,58	14,33	10,54	36,35	43,32	39,72
U-Net	4,83	15,13	12,36	15,99	13,09	49,76	53,08	49,23
SegNet	5,56	17,17	14,11	20,04	14,84	58,44	64,17	58,02
RefineNet	6,47	18,41	14,59	18,84	16,69	56,57	64,06	60,73
BiSeNet	4,05	11,66	8,48	10,89	8,42	32,45	35,78	30,98
PSPNet	7,05	11,57	21,62	24,06	21,45	87,90	82,21	80,22
DeepLab-v3	4,84	14,11	8,57	13,19	20,13	32,22	40,21	74,81
DeepLab-v3+	5,04	7,35	10,33	14,50	22,32	37,69	44,10	80,71
DenseASPP	6,35	11,58	16,89	20,53	18,11	69,30	68,09	67,02



**Figura 5.39** – Tempo de inferência (segundos) e desempenho mIoU (%) no conjunto de três imagens aéreas de teste (144 blocos de 576 » 256).

### 5.3.1.4 Testes adicionais com diferentes hiperparâmetros de treinamento

Para complementar nossa análise, treinamos o modelo BiSeNet (Xception) na configuração AT/DA, variando alguns hiperparâmetros, como tamanho de lote (*batch*) e tamanho dos blocos de treinamento. Dividimos os experimentos em duas estratégias principais: variando o tamanho do lote (três primeiras linhas da [Tabela 5.10](#)) e variando o tamanho dos blocos de treinamento (primeira linha e duas últimas).

**Tabela 5.10** – *Treinamento da rede BiSeNet AT(5)/DA com diferentes estratégias.*

Modelo de Segmentação	Entrada » Subamostragem	Batch	Blocos de train   val	Passos train   val por época	Tempo de treinamento
BiSeNet	576 » 256	8	288   144	36   18	2 h 59 min
(Xception)	576 » 256	2	288   144	144   72	4 h 15 min
26.081.488	576 » 256	16	288   144	18   9	+ 3 h 29 min
	288 » 256	8	1152   576	144   72	11 h 20 min
	576 » 512	8	288   144	36   18	*13 h 57 min

\*GPU de menor velocidade, +outro processo rodando na mesma GPU

1ª Estratégia – variando o tamanho de *batch*, com tamanho de bloco fixo de 576 » 256 *pixels* (subamostrado):

1. Treinamento padrão com *batch* = 8 (linha em azul na [Tabela 5.10](#), já avaliado nas [Seções 5.3.1.1 a 5.3.1.3](#));  
Tem 288 blocos de treinamento com 36 passos de atualização de pesos por época.
2. Treinamento com número menor de *batch* = 2;  
Tem mesmo número de parâmetros e blocos de treinamento que o anterior, mas o tempo de treinamento é maior, com o aumento do número de passos por época.  
Tem 288 blocos de treinamento com 144 passos por época.
3. Treinamento com número maior de *batch* = 16.  
Tem o menor número de passos por época de treinamento (18 passos). O tempo de treinamento deveria ser menor que *batch* = 16, mas foi maior devido a outro processo rodando simultaneamente na GPU.

2ª Estratégia – variando o tamanho dos blocos de treinamento, com *batch* fixo de 8:

4. Treinamento com blocos (menores) de 288 » 256 *pixels* sem subamostragem;  
O tempo de treinamento aumenta com o maior número de blocos de entrada.  
Tem 1.152 blocos de treinamento com 144 passos por época.
5. Treinamento com blocos (maiores) de 576 » 512 sem subamostragem.  
Tem 288 blocos de treinamento com 36 passos por época.  
Nesta opção, mesmo tendo o mesmo número de blocos e passos da 1ª estratégia (linha 1), o tempo de treinamento aumenta com o aumento do número de operações executadas em blocos maiores.

Um recorte menor sobre a imagem aérea original gera uma quantidade maior de blocos de treinamento, portanto, uma configuração 288 » 256 (linha 4 da [Tabela 5.10](#)) leva mais tempo de treinamento do que uma configuração 576 » 256 (linha 1), apesar de terem o mesmo tamanho de bloco na entrada da rede.

O tamanho efetivo dos blocos na entrada da rede afeta o tempo de treinamento, por causa do número de operações executadas em blocos maiores ou menores, como pode ser observado na configuração sem subamostragem 576 » 512 (linha 5) e com subamostragem 576 » 256 (linha 1 da [Tabela 5.10](#)). Sem subamostragem, o tempo de treinamento é maior.

Como os pesos são atualizados a cada lote de imagens, o tamanho do lote afeta o número de passos de atualização dos pesos, por época de treinamento, dado pela razão do número de blocos de treinamento pelo tamanho do lote ([Equação 5.1](#)). Assim, uma configuração 576 » 256 com lote menor ( $batch = 2$ ), na linha 2 da [Tabela 5.10](#), leva mais tempo do que com  $batch = 8$  (linha 1 da [Tabela 5.10](#)).

$$\text{Número de passos por época} = \frac{\text{número de blocos de treinamento}}{\text{tamanho do } batch} \quad (5.1)$$

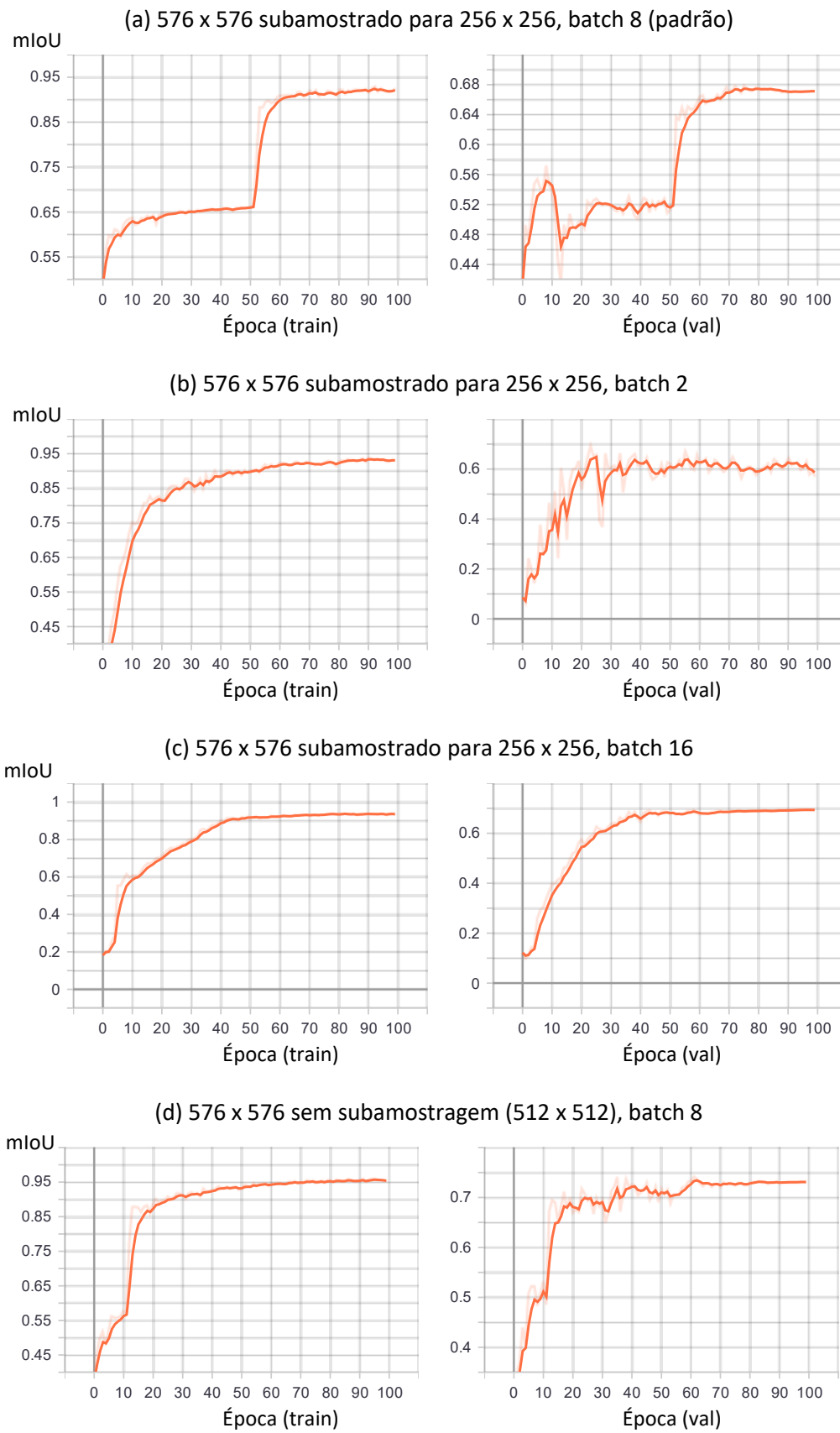
A [Figura 5.40](#) mostra os gráficos de desempenho de treinamento e validação de algumas estratégias da [Tabela 5.10](#), visualizados no Tensorflow. Na [Figura 5.40d](#), a rede treinada com blocos 576 » 512 (sem subamostragem) apresenta maior desempenho de treinamento e validação do que com entrada 576 » 256 com subamostragem ([Figura 5.40a](#)). Em outras palavras, a redução de tamanho na entrada da rede pode degradar o desempenho ao descartar informações espaciais cruciais para distinguir cultura e daninhas, no entanto, o desempenho de teste indica que a subamostragem em imagens de alta resolução aumenta o desempenho (linha 5 e 1 da [Tabela 5.11](#)), devido ao maior campo receptivo da rede. Como o tempo de treinamento sem subamostragem foi substancialmente maior do que com subamostragem (linha 5 e 1 da [Tabela 5.10](#), respectivamente), subamostramos a entrada da rede para acelerar o treinamento neste trabalho.

Além disso, a [Figura 5.40c](#) mostra que um tamanho maior de lote apresenta atualizações de pesos mais estáveis e suaves do que com lotes menores. Porém, tem menor desempenho de teste do que com  $batch = 8$  (linha 3 e 1 da [Tabela 5.11](#)).

A [Tabela 5.11](#) mostra os resultados de desempenho mIoU de validação e teste de todos os experimentos, usando os pesos obtidos no final do treinamento. Segundo [[DEEPLARNINGBOOK](#)], embora seja difícil fazer afirmações gerais apenas com estes experimentos sobre os efeitos da alteração de hiperparâmetros, que variam dependendo do conjunto de dados e do modelo de rede, é bem conhecido que tamanhos maiores de lotes levam a um *gap* de generalização maior (diferença entre o desempenho de treinamento e teste), ou seja, uma precisão menor nos dados de teste, conforme comprovado por Monteiro *et. al* [[MONTEIRO 2019](#)]. Como o tamanho de  $batch = 2$  leva mais tempo para treinamento e obteve o menor desempenho de validação e teste, escolhemos um tamanho de lote padrão ( $batch = 8$ ) para treinar todos os modelos em menor tempo de duração.

**Tabela 5.11** – Desempenho de BiSeNet AT(5)/DA com diferentes estratégias de treinamento.

Modelo de Segmentação	Entrada » Subamostragem	Batch	mIoU (%)	
			Val	Test
BiSeNet (Xception)	576 » 256 <sup>(8)</sup>	8	67,12	<b>68,91</b>
	576 » 256 <sup>(2)</sup>	2	57,04	61,52
	576 » 256 <sup>(16)</sup>	16	69,33	64,17
Parâmetros 26.081.488	288 » 256 <sup>(8)</sup>	8	72,15	66,43
	576 » 512 <sup>(8)</sup>	8	73,10	66,54



**Figura 5.40** – Tensorboard: desempenho mIoU de treinamento e validação da rede BiSeNet (Xception) AT/DA com diferentes tamanhos de entrada e batch.

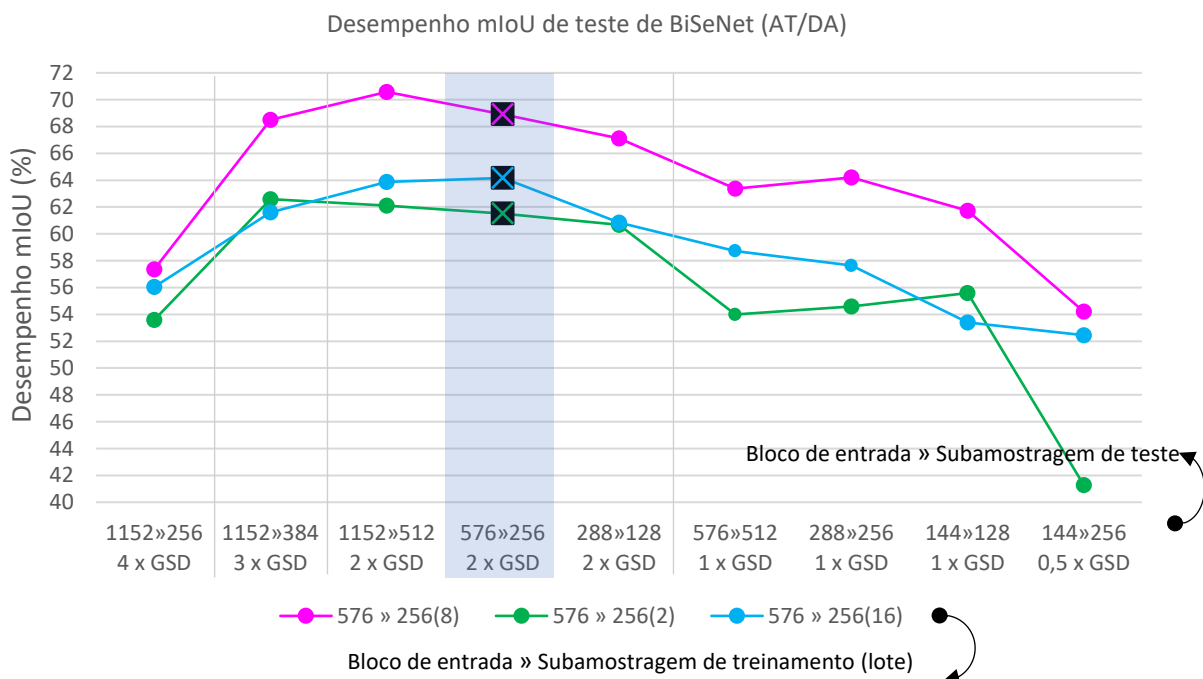


Analizamos também o desempenho de cada um destes experimentos em diferentes tamanhos de blocos de teste e GSD, como mostra a [Tabela 5.12](#).

**Tabela 5.12** – Desempenho *mIoU* de cada estratégia de treinamento da rede *BiSeNet AT(5)/DA* em diferentes resoluções dos blocos de teste.

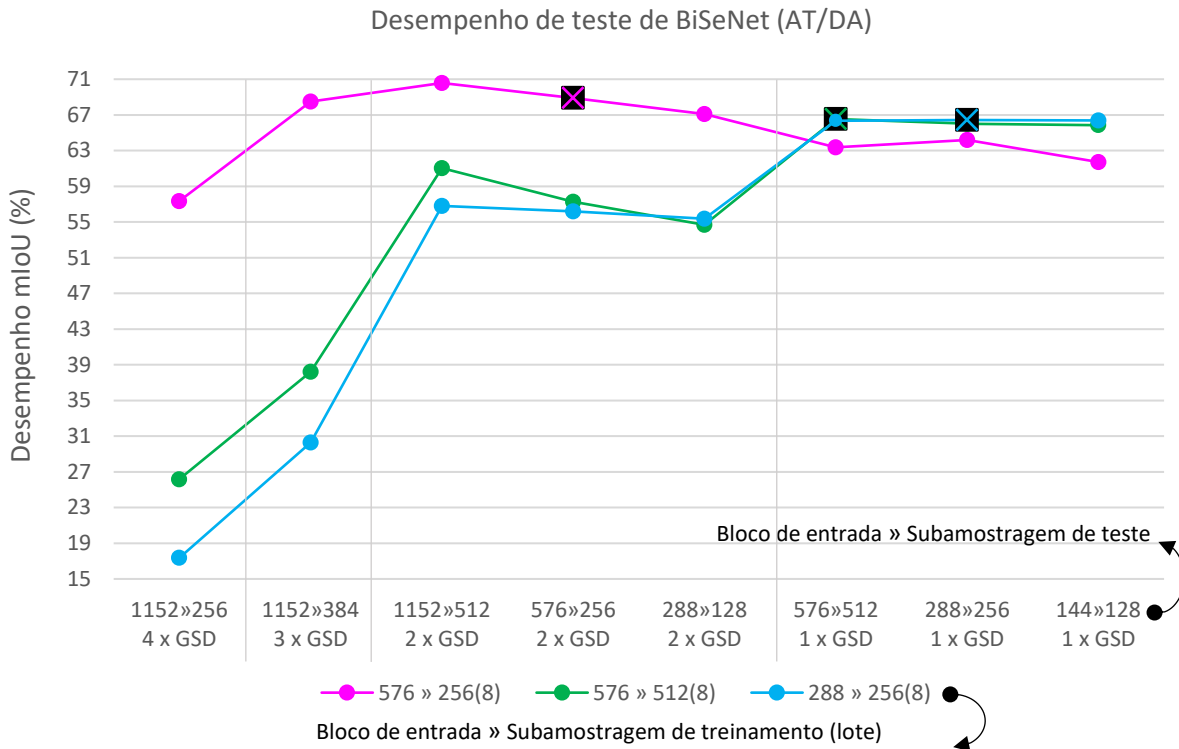
Entrada de Treinamento BiSeNet (Xception)	mIoU (%) de 3 imagens RGB de teste							
	1152 » 256	1152 » 384	1152 » 512	576 » 256	288 » 128	576 » 512	288 » 256	144 » 128
	36 blocos 4cm/px	36 blocos 3cm/px	36 blocos 2cm/px	144 blocos 2cm/px	576 blocos 2cm/px	144 blocos 1cm/px	576 blocos 1cm/px	2304 blocos 1cm/px
576 » 256 (8)	57,35	68,49	70,58	68,91	67,11	63,38	64,21	61,72
576 » 256 (2)	53,60	62,59	62,11	<b>61,52</b>	60,68	53,99	54,59	55,59
576 » 256 (16)	56,04	61,61	63,88	<b>64,17</b>	60,83	58,73	57,64	53,39
288 » 256 (8)	17,36	30,31	56,82	56,19	55,37	66,35	<b>66,43</b>	66,38
576 » 512 (8)	26,18	38,22	61,04	57,26	54,70	<b>66,54</b>	66,01	65,85

O gráfico da [Figura 5.41](#) mostra que o treinamento da rede *BiSeNet* (AT/DA) usando a primeira estratégia tem melhor desempenho com tamanho de *batch* = 8 em todas as resoluções espaciais dos blocos de teste.



**Figura 5.41** – Comparação da rede *BiSeNet* treinada com tamanho de *batch* diferentes.

Em seguida, fixamos o tamanho de *batch* = 8 e treinamos com recortes de blocos de 576 » 512 e 288 » 256 *pixels* (sem subamostragem). Em imagens de alta resolução, blocos com tamanho de recorte menor contém uma pequena área de informação de contexto sobre o terreno (sem informações sobre áreas na vizinhança da planta), podendo degradar o desempenho de teste. Blocos maiores requerem mais recursos de memória para processamento do treinamento. O gráfico da [Figura 5.42](#) apresenta os desempenhos de teste da rede *BiSeNet* (AT/DA) com diferentes tamanhos de blocos e GSD.



**Figura 5.42** – Comparação da rede BiSeNet treinada em diferentes tamanhos de blocos.

O gráfico da [Figura 5.42](#) tem três marcadores quadrados pretos com valores de desempenho semelhantes. Cada marcador indica o desempenho de uma rede treinada com o mesmo tamanho e GSD de teste. Isto mostra que o treinamento da rede BiSeNet com blocos 576 » 512 (marcador sobre linha verde) ou 288 » 256 (marcador sobre linha azul) – ambos sem subamostragem e com resolução espacial de aproximadamente 1 x GSD – tem desempenho similar ao do treinamento com blocos subamostrados de 512 » 256 (marcador sobre linha rosa), com resolução de aproximadamente 2 x GSD. Entretanto, esta pequena diferença de desempenho ocorre apenas quando o GSD e o tamanho do bloco de teste são semelhantes ao do treinamento correspondente. Como pode-se observar no gráfico, quanto mais distante o GSD do bloco de teste em relação ao GSD de treinamento, menor é o desempenho de generalização.

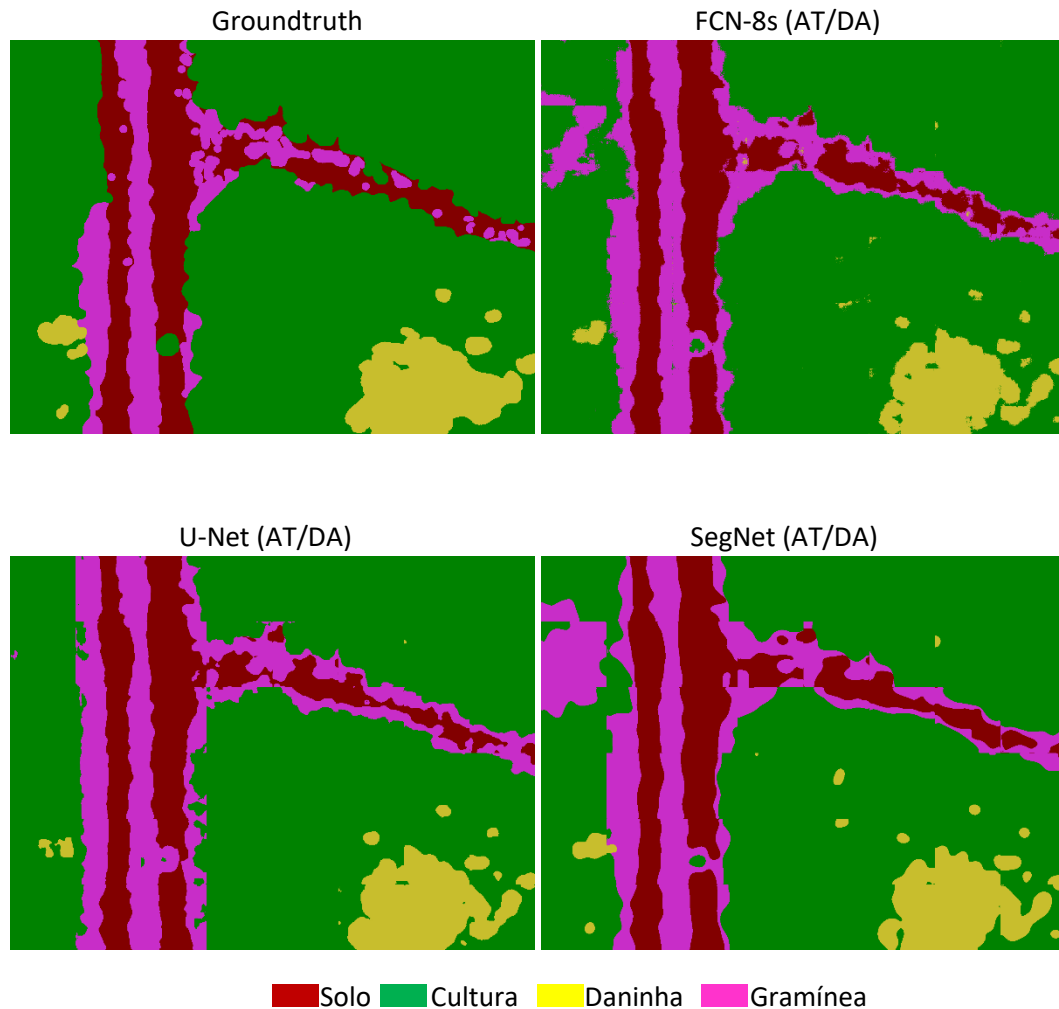
Este problema afeta o desempenho da segmentação semântica, especialmente em mosaico de imagens aéreas de alta resolução. Por exemplo, durante a aquisição das imagens de VANT, a altura de voo (distância em relação ao solo) varia de acordo com o relevo, como mostrado na [Figura 4.35](#). Portanto, o GSD varia em cada região do mosaico, gerando objetos em escalas variadas, o que acaba afetando o desempenho da predição dos modelos de rede que utilizam escala única.

Além disso, em uma aplicação real de segmentação semântica usando mosaico de imagens aéreas de um campo agrícola, o GSD médio do mosaico na entrada da rede pode ter um valor diferente do treinado pela rede, dependendo da altura de cada voo e do sensor utilizado. Conseqüentemente, a escolha ideal do GSD do bloco de treinamento deve levar em consideração os requisitos de cada aplicação, para que seja o mais próximo possível do GSD treinado pela rede.

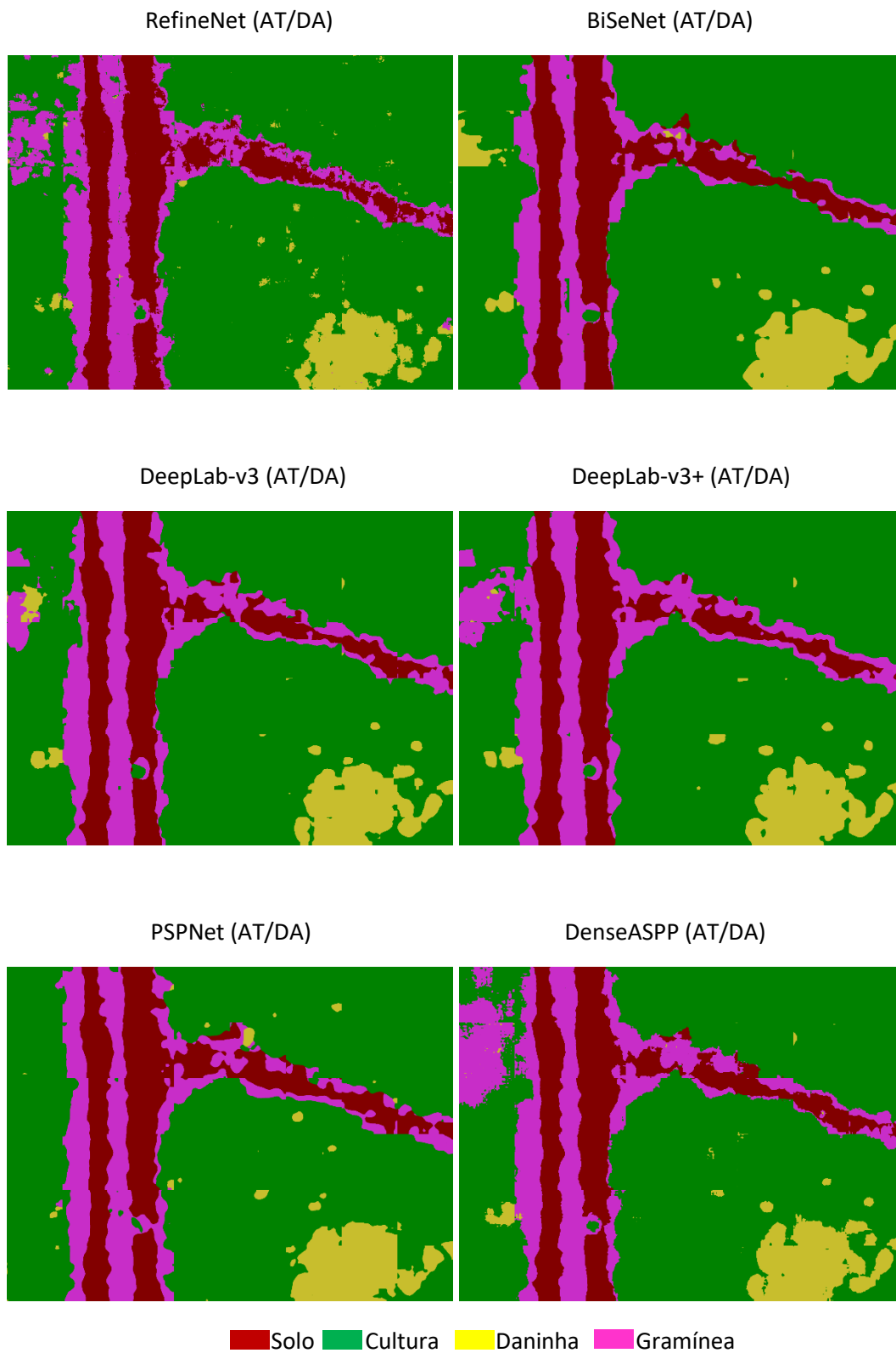
### 5.3.2 Resultados qualitativos com imagens de teste RGB

Para análise qualitativa da predição de cada modelo de rede (treinada na configuração AT/DA com blocos de  $576 \times 256$  pixels) a partir de blocos de teste de  $576 \times 256$  pixels, apresentamos o *groundtruth* de uma imagem aérea RGB de teste e as imagens preditas por cada rede (Figura 5.43). As predições subamostradas de  $2.048 \times 1.536$  pixels foram obtidas após composição da grade de  $8 \times 6$  blocos preditos de  $256 \times 256$  pixels na saída da rede.

Considerando o pequeno tamanho do conjunto de dados de treinamento, ainda assim, a maioria dos modelos de rede conseguiu gerar bons mapas de probabilidade para segmentação semântica, com classificação densa dos pixels das imagens aéreas RGB em quatro classes, confirmando os resultados quantitativos da Tabela 5.6 ou Tabela 5.7, com pouca diferença de desempenho entre as redes. Todas as redes foram capazes de discriminar entre plantas daninhas e cultura de cana-de-açúcar, além de localizar as áreas de solo exposto e gramíneas. Em conformidade com os resultados quantitativos, o modelo RefineNet gerou os piores resultados qualitativos.



**Figura 5.43** – Comparação do *groundtruth* com a predição da imagem 10 de teste, prevista pelos modelos de segmentação treinados e testados com blocos  $576 \times 256$  ( $2 \times$  GSD).



**Figura 5.43** – *Continuação da figura anterior.*

Ao analisar visualmente os dados em cada experimento, avaliamos as previsões em relação aos blocos corretos com ênfase na classe daninha, ou seja, blocos em que todas as plantas daninhas foram corretamente atribuídas à classe correta (Figura 5.44a). As previsões incorretas também foram analisadas, como:

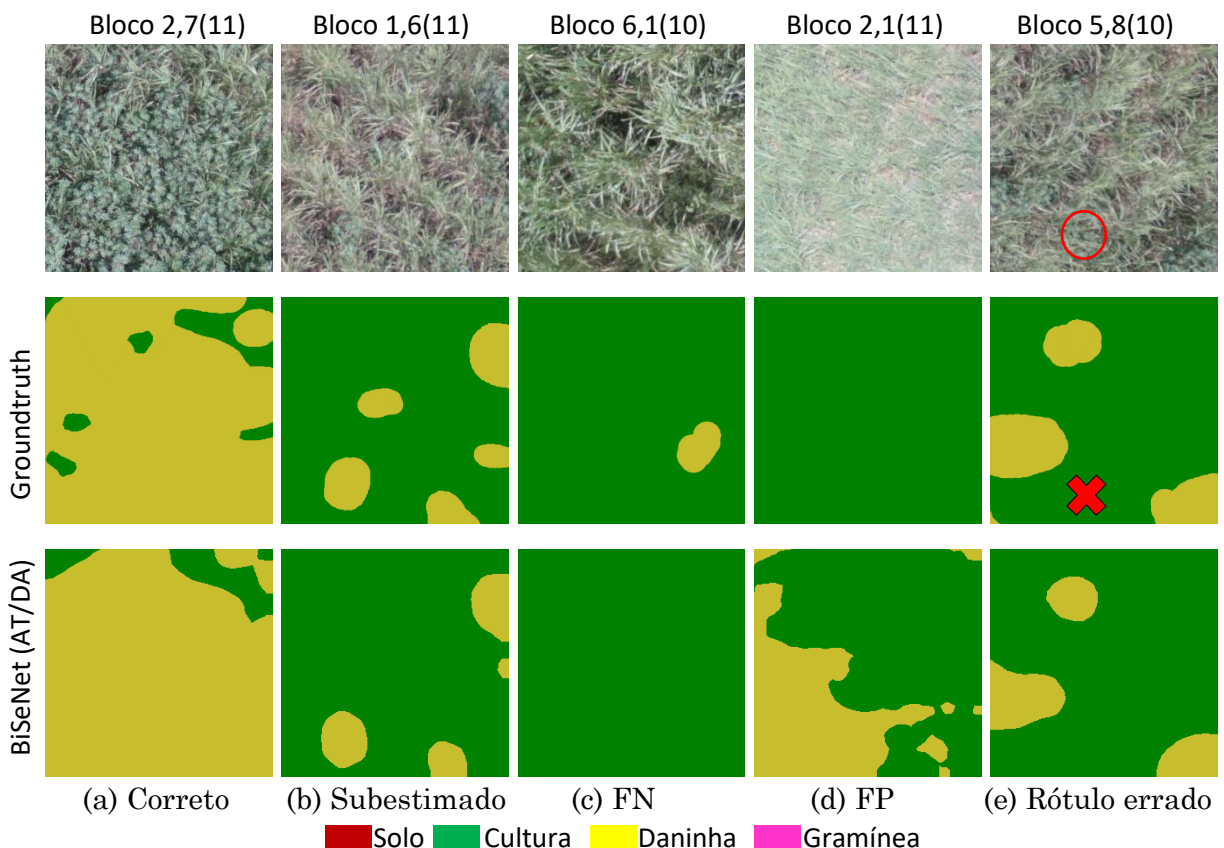
(1) subestimada, ou seja, blocos infestados de ervas daninhas nos quais algumas daninhas foram detectadas, mas outras permaneceram não detectadas (Figura 5.44b);

(2) falso negativo, ou seja, blocos com ervas daninhas, nos quais nenhuma daninha foi detectada (Figura 5.44c);

(3) falso positivo, ou seja, blocos em que as ervas daninhas foram superestimadas (por exemplo, cultura classificada como daninha) (Figura 5.44d).

Alguns rótulos do *groundtruth* foram anotados incorretamente, ou seja, daninha rotulada como cultura ou vice-versa, devido à dificuldade de rotulação manual e semelhança espectral entre as plantas (Figura 5.44e).

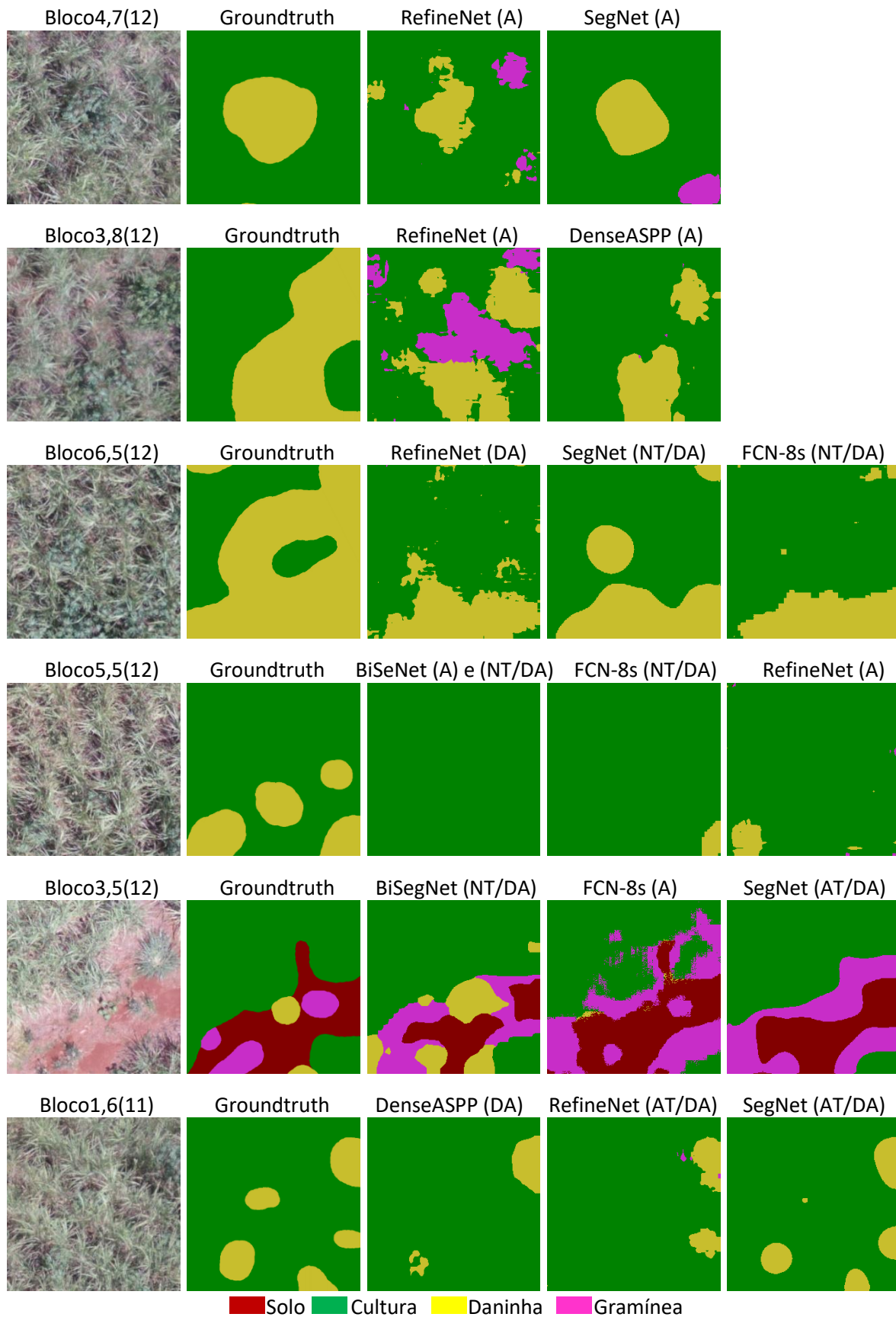
Do ponto de vista do manejo de plantas daninhas, os erros falso positivos (superestimação de daninhas) são mais aceitáveis do que os erros falso negativos (não detecção de daninhas) ao gerar mapas de distribuição de plantas daninhas em uma cultura agrícola, considerando que é melhor tratar áreas livres de daninhas em vez de correr o risco de não tratar as daninhas [PEÑA *et al.* 2015].



**Figura 5.44** – Exemplos de previsão da rede BiSeNet (AT/DA) a partir de quatro blocos rotulados das imagens de teste 10 e 11, mostrando: (a) classificação correta; (b) subestimação de ervas daninhas; (c) falso negativo (não detecção de ervas daninhas); (d) falso positivo (superestimação de ervas daninhas); (e) erro na rotulação manual.



A [Figura 5.45](#) compara outros exemplos de blocos, onde os modelos de redes em diferentes configurações falharam na segmentação semântica dos *pixels*.

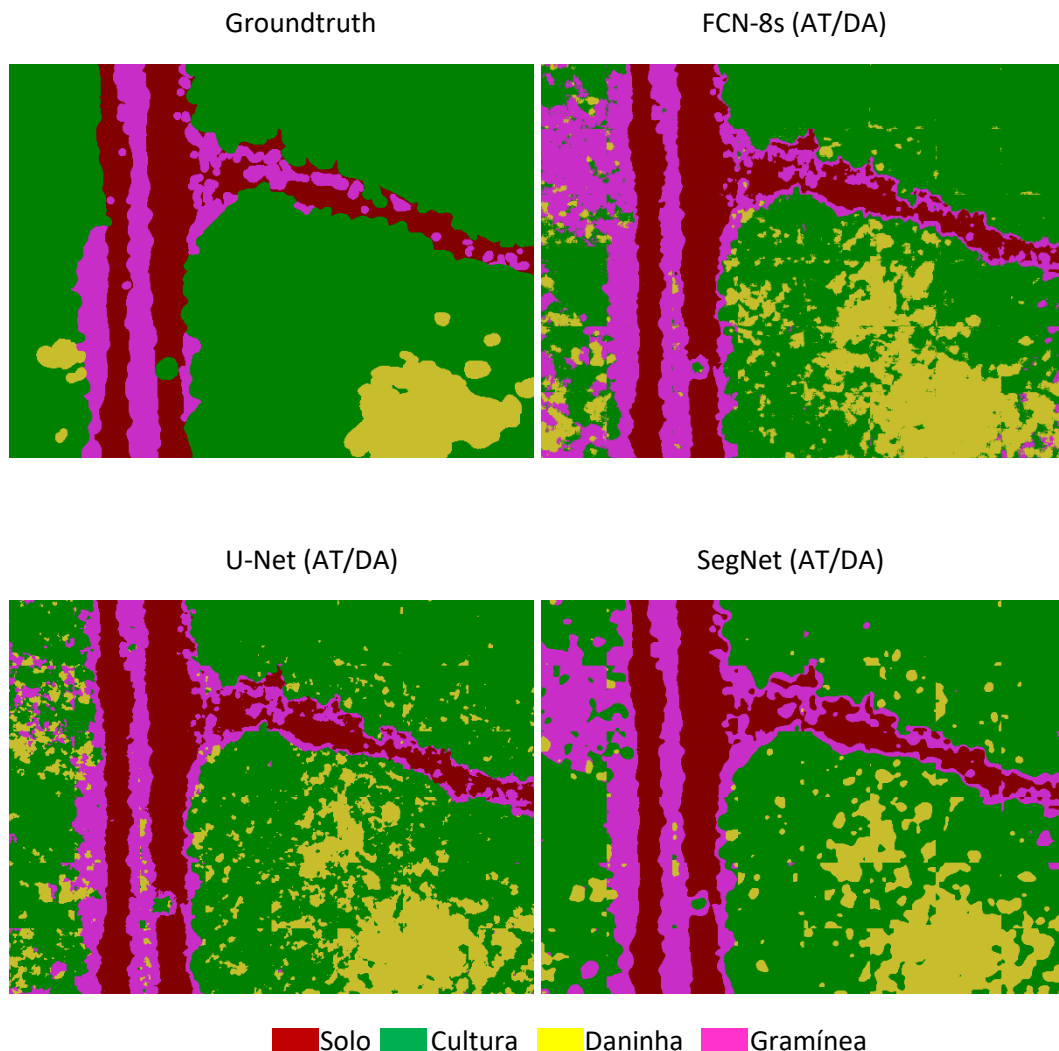


**Figura 5.45** – Exemplos de previsões que falharam em blocos da imagem original.

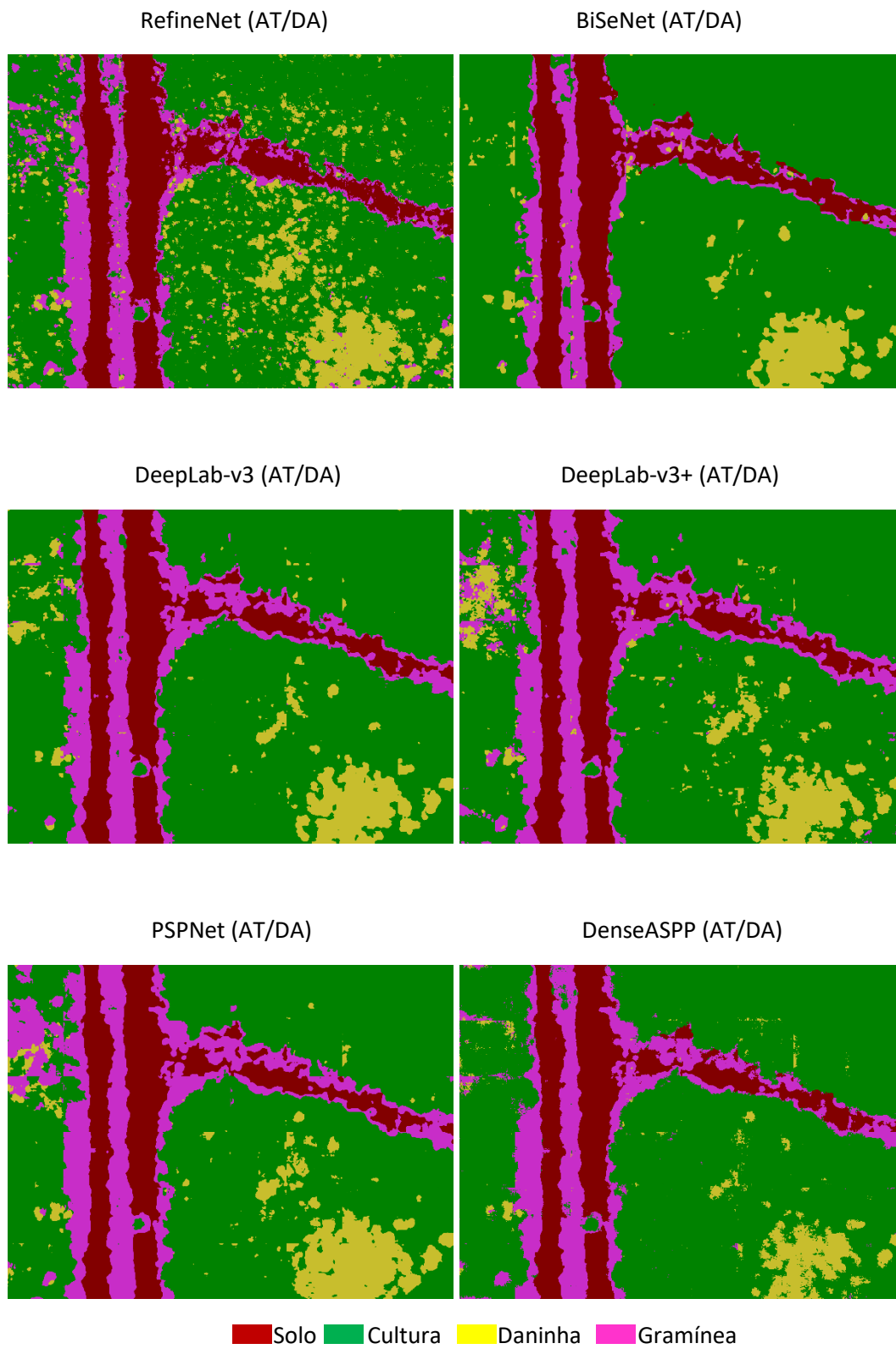
### 5.3.2.1 Teste com tamanhos de blocos e GSD diferentes do treinamento

Os resultados qualitativos da [Figura 5.43](#) confirmam que a maioria dos modelos de rede tem bom desempenho quando o treinamento e teste são executados com mesmo tamanho de bloco e GSD, com pouca diferença de desempenho entre os modelos, como pode ser verificado na coluna azul da [Tabela 5.8](#) e da [Figura 5.37](#). No entanto, ao analisar o desempenho quando o tamanho do bloco e o GSD de teste são diferentes do treinamento, a diferença de desempenho entre os modelos aumenta.

A [Figura 5.46](#) mostra o resultado visual da predição da imagem 10 de teste, usando as redes treinadas com blocos de 576 » 256 (2 x GSD) e testadas com blocos de 576 » 512 (1 x GSD) das imagens aéreas RGB, que corresponde ao desempenho da coluna 576 » 512 da [Tabela 5.8](#) e [Figura 5.37](#). Os modelos de segmentação BiSeNet e DeepLab-v3 com atributos multiescala tiveram um resultado visual mais satisfatório do que as redes em escala única.



**Figura 5.46** – Comparação do *groundtruth* com a predição da imagem 10 de teste, gerada pelos modelos de segmentação treinados com blocos 576 » 256 (2 x GSD) e testadas com blocos de 576 » 512 (1 x GSD).

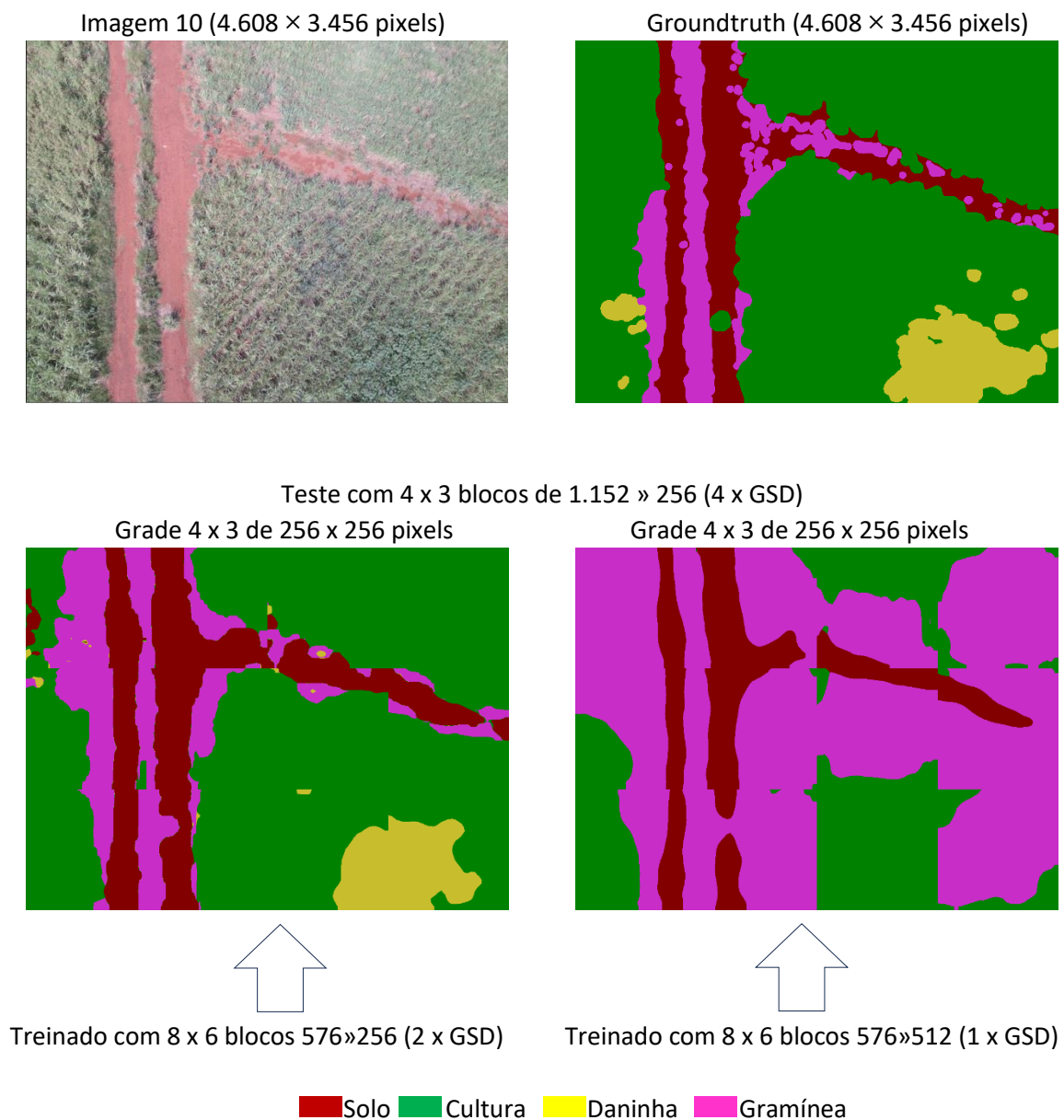


**Figura 5.46** – *Continuação da figura anterior.*

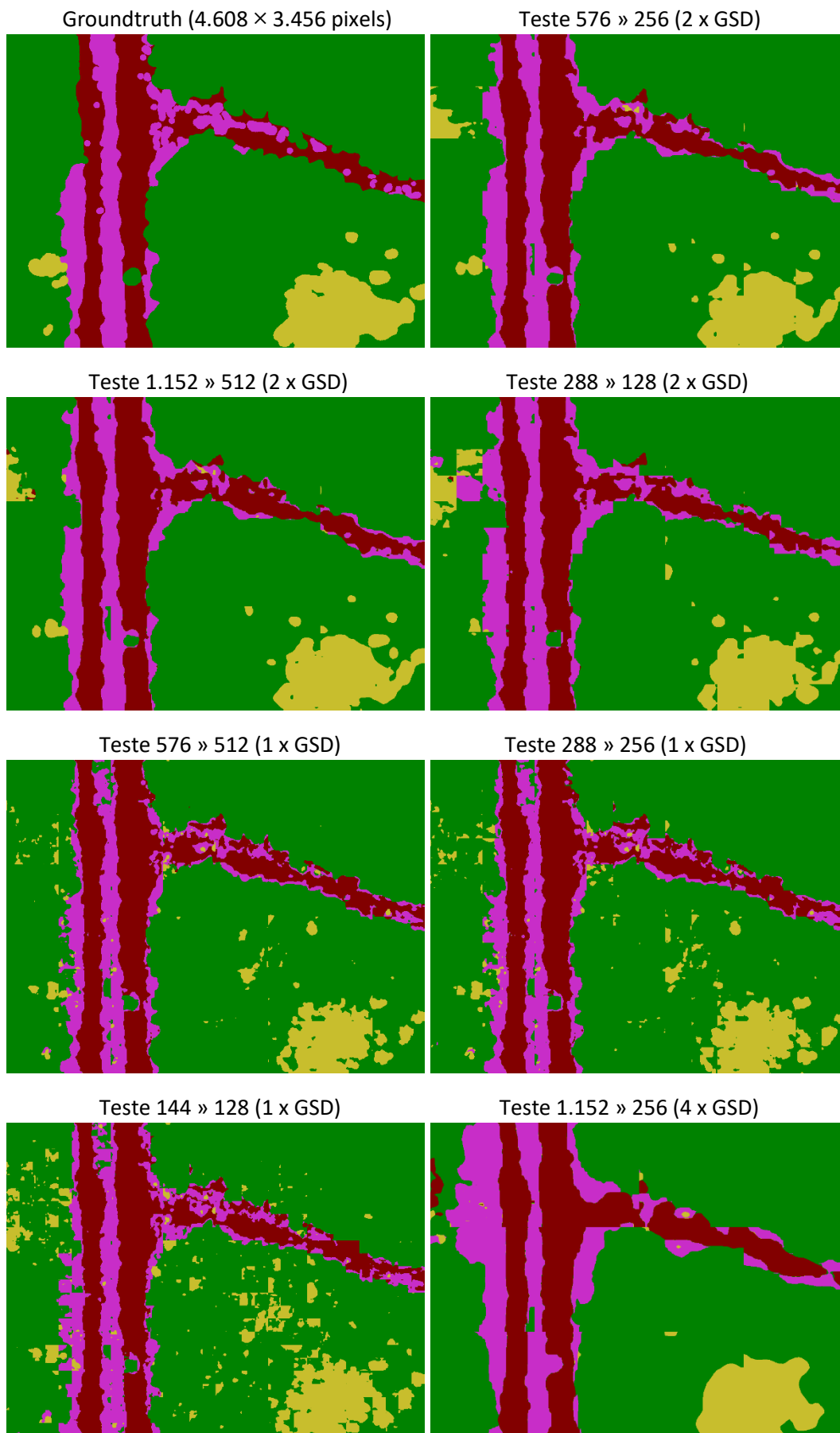
### 5.3.2.2 Treinamento com diferentes tamanhos de blocos e GSD

O gráfico da [Figura 5.42](#) mostra que quanto mais distante o GSD dos blocos de teste em relação ao GSD dos blocos de treinamento, menor será o desempenho. Por exemplo, o desempenho da rede BiSeNet treinada com blocos de 576 » 512 (1 x GSD) e testada com blocos de 1.152 » 256 (4 x GSD) é menor do que o desempenho da rede treinada com blocos de 576 » 256 (2 x GSD) e testada com resolução de 4 x GSD, como mostra a [Figura 5.47](#).

A [Figura 5.48](#) e a [Figura 5.49](#) mostram os resultados da rede BiSeNet (AT/DA) treinadas com blocos de 576 » 256 (2 x GSD) e 576 » 512 (1 x GSD), respectivamente, e testadas com várias resoluções GSD e tamanhos de blocos, correspondentes ao resultado quantitativo da [Figura 5.42](#).

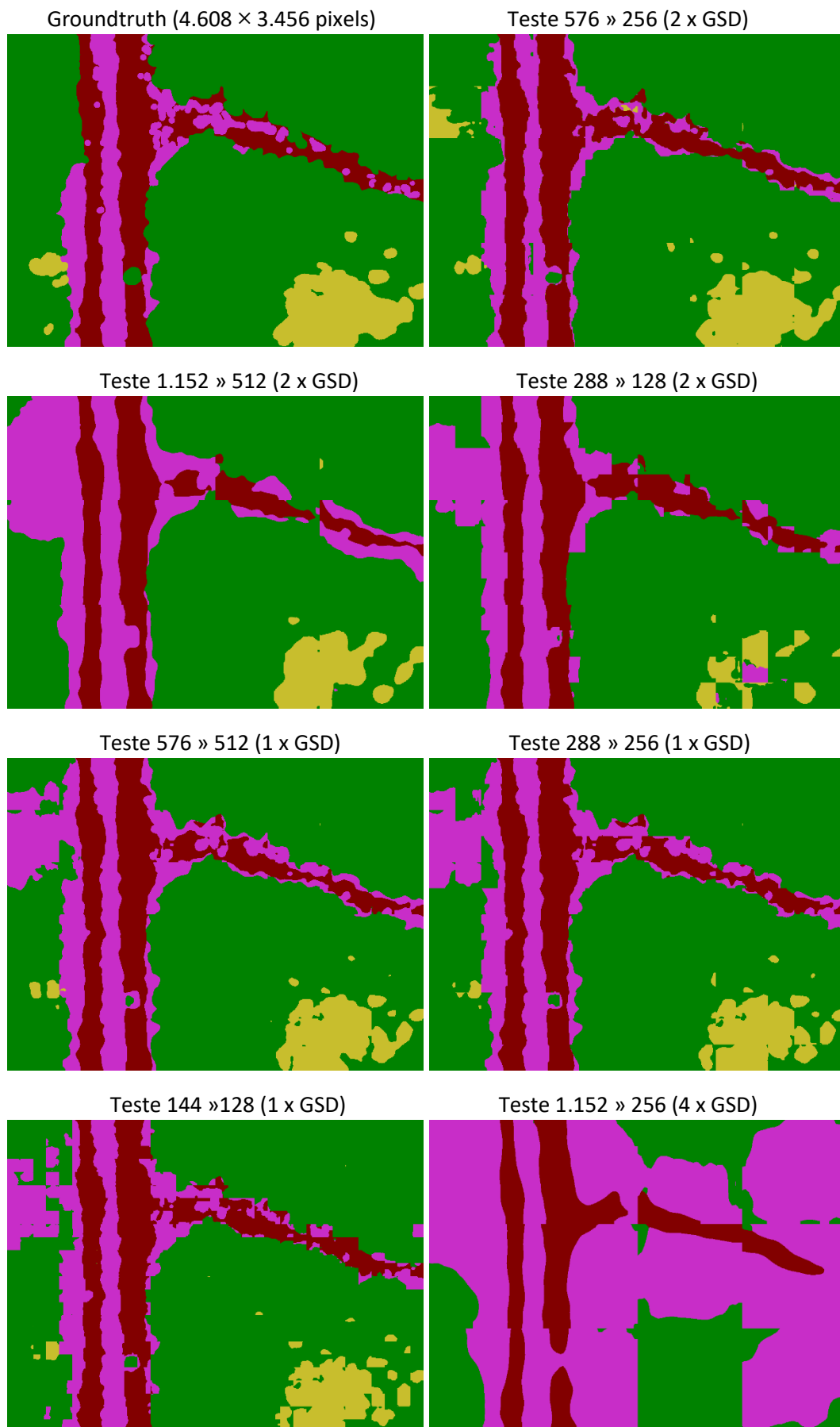


**Figura 5.47** – Modelo BiSeNet treinado com blocos 576 » 256 (2 x GSD) ou blocos 576 » 512 (1 x GSD), e testado com blocos 1.152 » 256 (4 x GSD).



**Figura 5.48** – Modelo BiSeNet treinado com 576 » 256 (2 x GSD).





**Figura 5.49** – Modelo BiSeNet treinado com 576 » 512 (1 x GSD).

### 5.3.3 Resultados quantitativos com mosaico de teste RGB

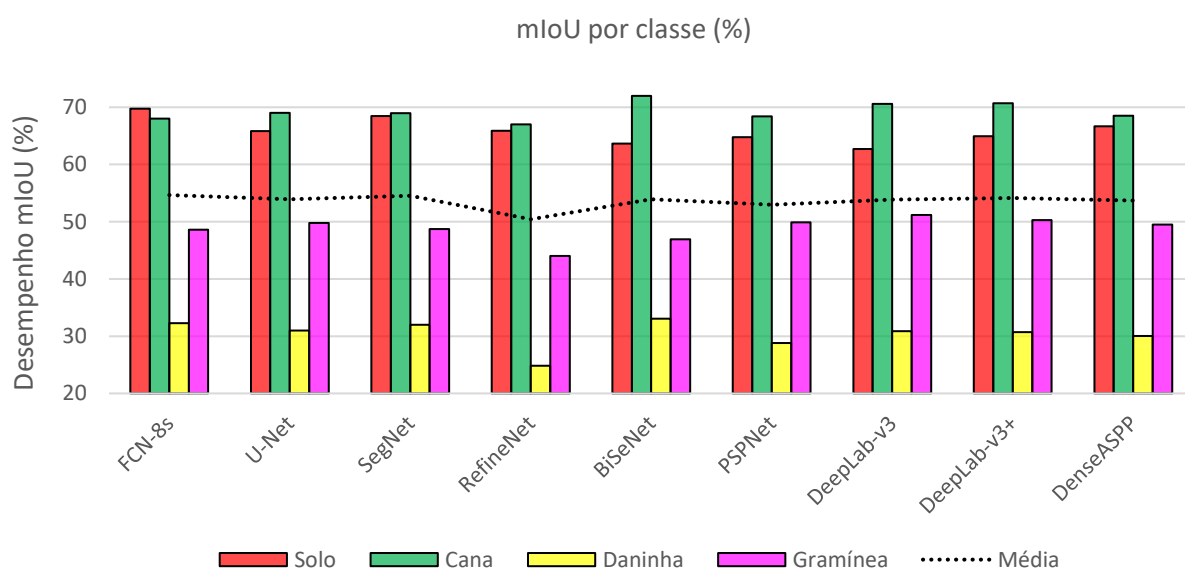
Para avaliar o desempenho das redes em uma aplicação real usando mosaico de imagens de VANT, a configuração AT/DA de cada modelo – que obteve melhor desempenho nas imagens aéreas de teste (na [Tabela 5.6](#)) – foi avaliada no conjunto de blocos do ortomosaico RGB, após a remoção dos blocos inválidos nas bordas do mosaico. Na [Tabela 5.13](#), os blocos de teste do mosaico subamostrados de 512 » 256 (2 x GSD) têm o mesmo tamanho de 576 » 256 (2,25 x GSD) dos blocos das imagens aéreas de treinamento.

Os resultados quantitativos na [Tabela 5.13](#) e no gráfico da [Figura 5.50](#) mostram que a rede BiSeNet obteve o melhor desempenho mIoU nas classes cana e daninha, destacados em negrito na tabela. Todas as redes tiveram resultados parecidos por classe, quando usado o mesmo GSD e o mesmo tamanho de bloco de treinamento e teste, como era esperado pelo resultado na coluna azul da [Figura 5.37](#).

**Tabela 5.13** – Desempenho das melhores configurações de redes de segmentação semântica por classe no conjunto de teste composto por blocos do ortomosaico RGB.

Modelo de Segmentação	CNN-base (backbone)	Solo mIoU %	Cana mIoU %	Daninha mIoU %	Gramínea mIoU %	mIoU médio %	Configuração de rede
FCN-8s	VGG-16	<b>69,77</b>	68,02	32,25	48,59	<b>54,66</b>	AT(5)/DA
U-Net	VGG-16	65,84	69,01	31,01	49,77	53,91	AT(5)/DA
SegNet	VGG-16	68,49	68,98	31,97	48,73	54,54	AT(5)/DA
RefineNet	ResNet-50	65,89	67,00	24,81	44,01	50,43	AT(5)/DA
BiSeNet	Xception	63,66	<b>71,98</b>	<b>33,04</b>	46,93	53,90	AT(5)/DA
PSPNet	ResNet-50	64,78	68,43	28,81	49,87	52,97	AT(3)/DA
DeepLab-v3	ResNet-50	62,69	70,61	30,88	<b>51,17</b>	53,84	AT(4)/DA
DeepLab-v3+	ResNet-50	64,93	70,68	30,71	50,27	54,15	AT(4)/DA
DenseASPP	ResNet-50	66,68	68,52	30,02	49,52	53,69	AT(3)/DA

DA – Data Augmentation, AT – All Trainable



**Figura 5.50** – Desempenho IoU por classe das melhores configurações (AT/DA) de cada modelo no conjunto de teste do mosaico.

### 5.3.4 Resultados qualitativos com mosaico de teste RGB

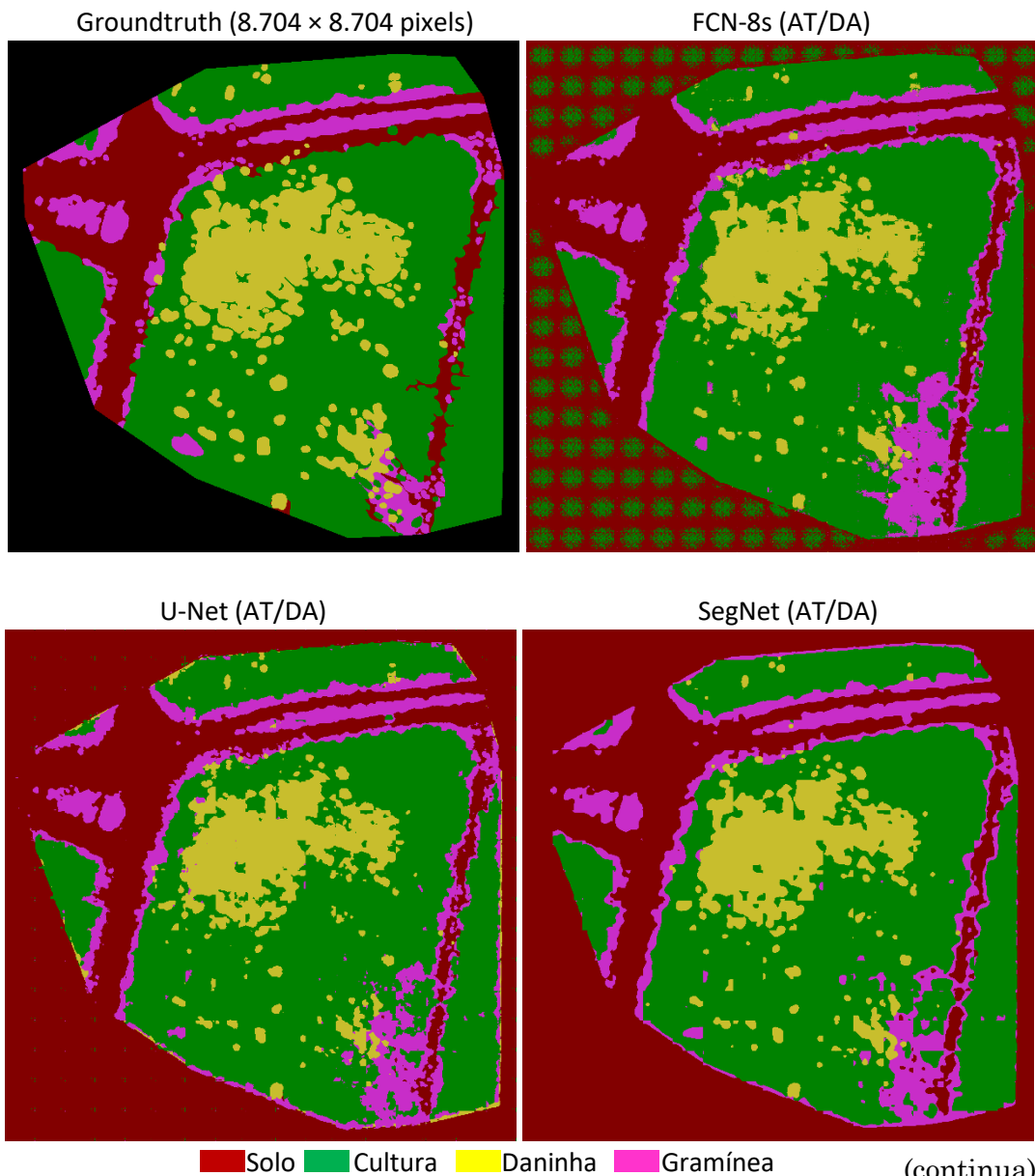
A [Figura 5.51](#) mostra os mosaicos previstos a partir dos blocos de teste do ortomosaico RGB de  $512 \times 256$  ( $2 \times \text{GSD}$ ), usando cada modelo de segmentação semântica treinado com blocos de imagens aéreas de  $576 \times 256$  ( $2,25 \times \text{GSD}$ ), correspondente ao resultado quantitativo da [Tabela 5.13](#).

Após a predição, a grade regular de  $17 \times 17$  blocos preditos de  $256 \times 256$  pixels é recomposta para retornar ao tamanho do mosaico subamostrado de  $4.352 \times 4.352$  pixels. Como podemos observar, os pixels nas bordas do mosaico não são corretamente preditos, uma vez que não houve treinamento da classe “fundo preto” correspondente. Desta forma, o mosaico previsto deve passar por um mascaramento de borda para ser comparado qualitativamente ao *groundtruth*.

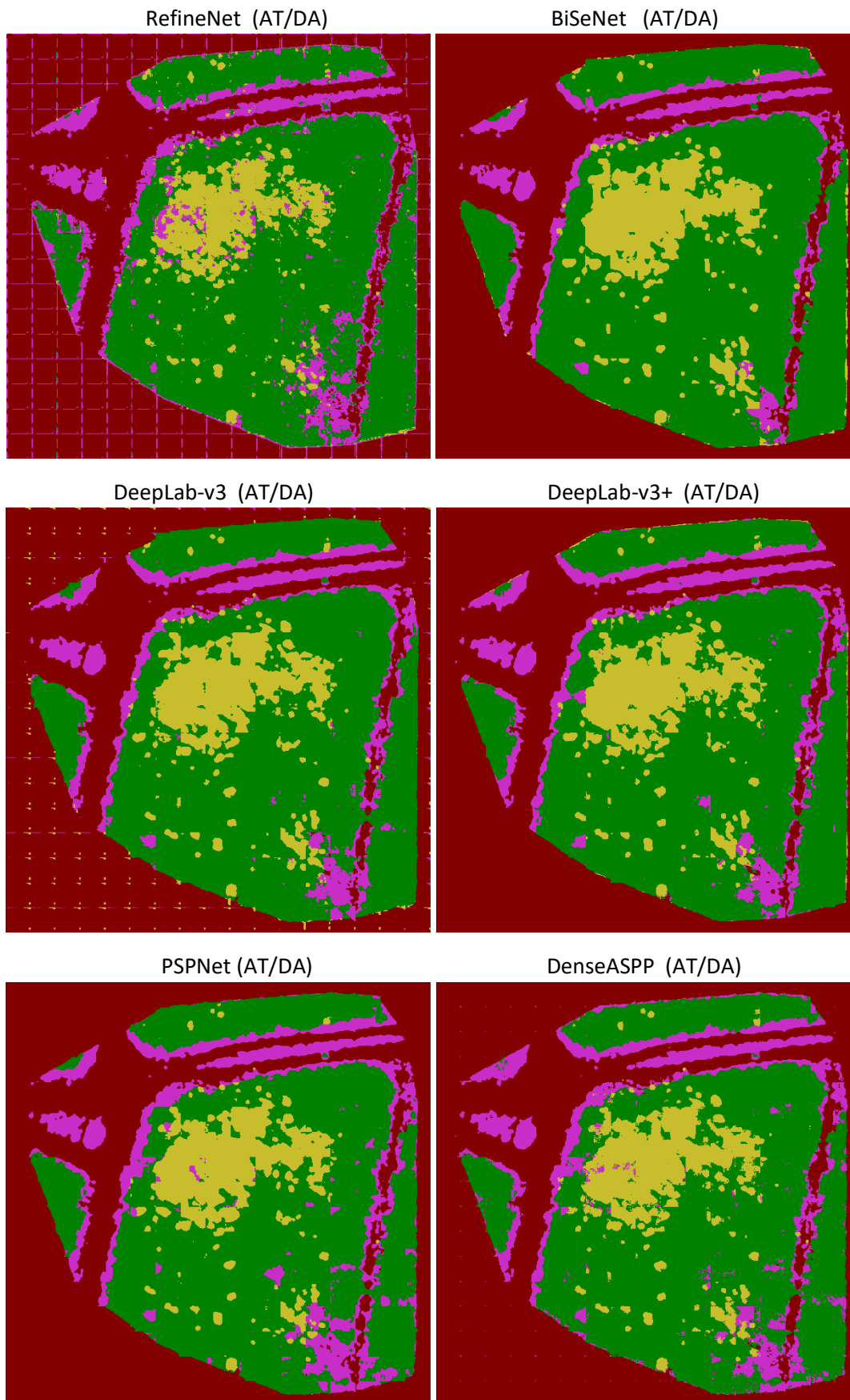
Treinamento:  $17 \times 17$  blocos de treinamento  $576 \times 256$  ( $2,25 \times \text{GSD}$ ) da imagem aérea RGB.

Entrada para predição:  $17 \times 17$  blocos de teste  $512 \times 256$  ( $2 \times \text{GSD}$ ) do mosaico RGB.

Saída prevista:  $17 \times 17$  blocos de  $256 \times 256$  pixels ( $4.352 \times 4.352$  pixels).



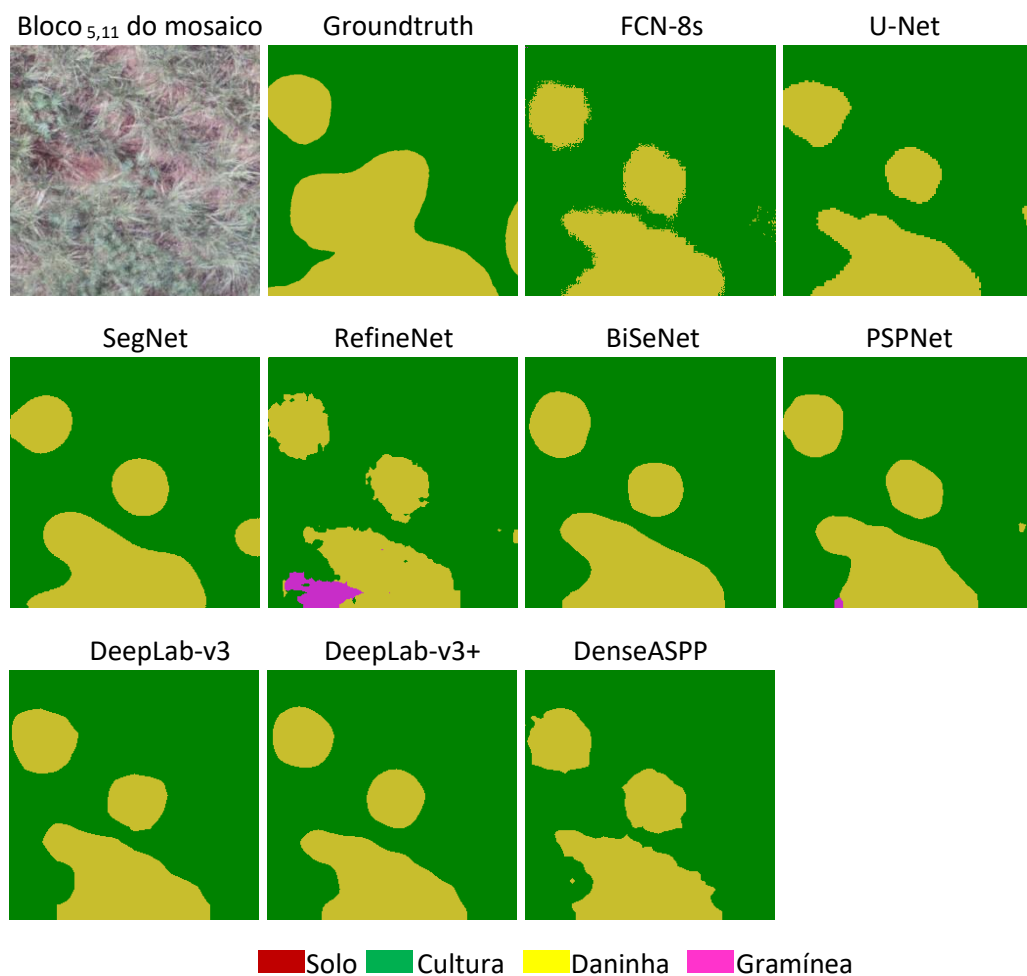
(continua)



**Figura 5.51** – *Predições sem máscara de borda, a partir dos blocos 512 » 256 (2 x GSD) do mosaico RGB, com redes treinadas com blocos de imagens aéreas 576 » 256 (2,25 x GSD).*

Vale lembrar que o mosaico é gerado pela projeção dos *pixels* das imagens originais, corrigidas geometricamente sobre um modelo digital de superfície (MDS), bem como por uma mesclagem das intensidades dos *pixels* de várias imagens aéreas sobrepostas, que alteram as características originais da imagem capturada pela câmera embarcada no VANT. Sendo assim, é esperado que o resultado de predição da rede sobre o conjunto de teste formado por blocos do ortomosaico seja inferior ao resultado sobre o conjunto de teste da imagem original. No entanto, a análise visual das predições da [Figura 5.51](#) mostra que todas as redes têm capacidade de fazer a segmentação semântica de plantas daninhas no mosaico de imagens aéreas RGB, apesar das distorções e mesclagem de *pixels*. Os melhores resultados qualitativos foram obtidos pelas redes BiSeNet e DeepLab-v3.

Os resultados da predição a partir do ortomosaico foram visualmente satisfatórios uma vez que as plantas daninhas foram localizadas corretamente, como mostra um exemplo na [Figura 5.52](#). No Bloco<sub>5,11</sub> do mosaico, apesar do *groundtruth* ter rótulos de classe com união de agrupamentos distintos de plantas daninhas, os modelos conseguiram separar corretamente as daninhas. Uma análise visual qualitativa mais detalhada será descrita na [Seção 7.1.2](#).

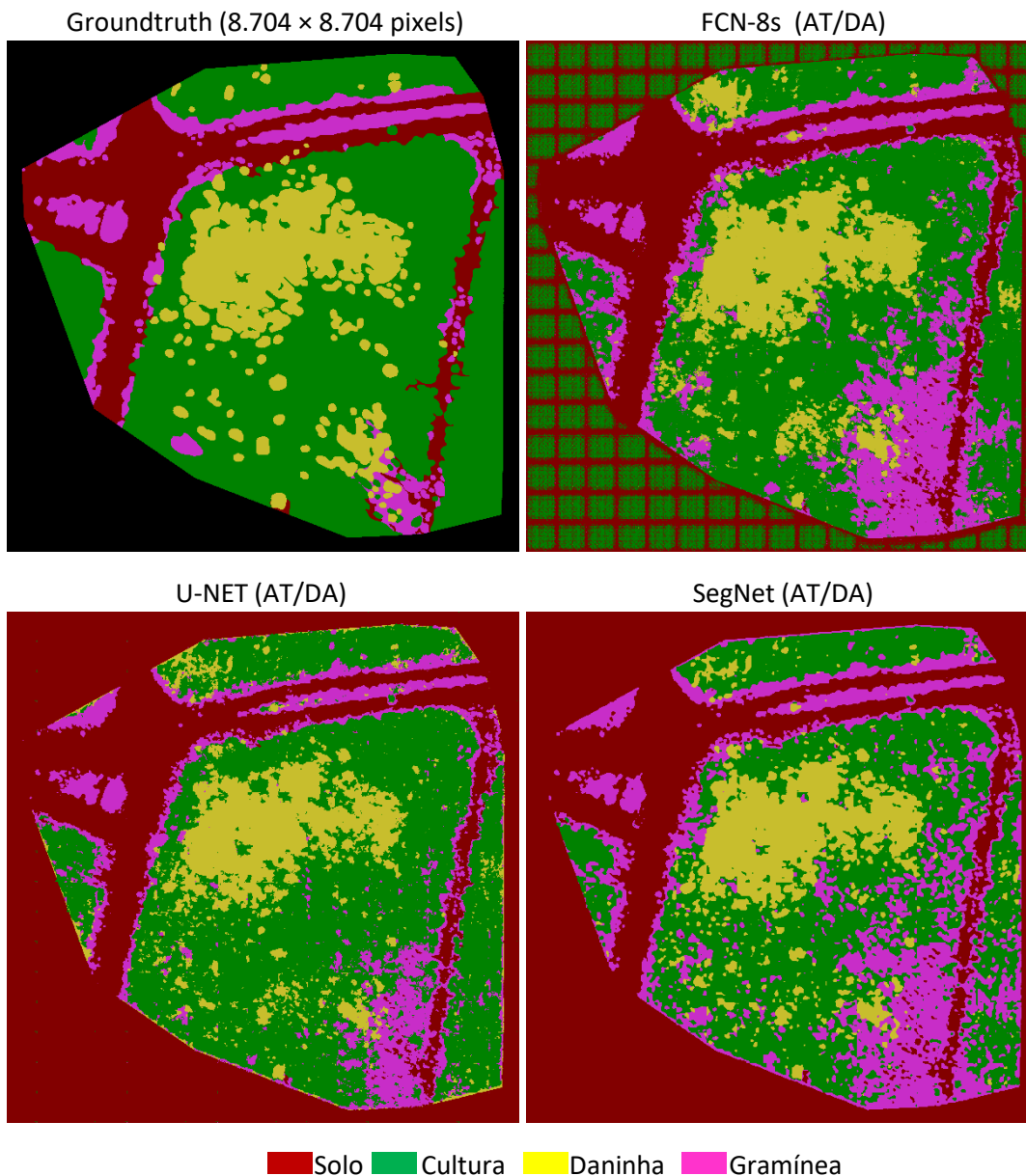


**Figura 5.52** – Bloco<sub>5,11</sub> do mosaico RGB; *groundtruth* de  $512 \times 512$  pixels; e predições de  $256 \times 256$  pixels das redes na configuração AT/DA, treinadas com blocos de imagens aéreas  $576 \gg 256$  ( $2,25 \times$  GSD).

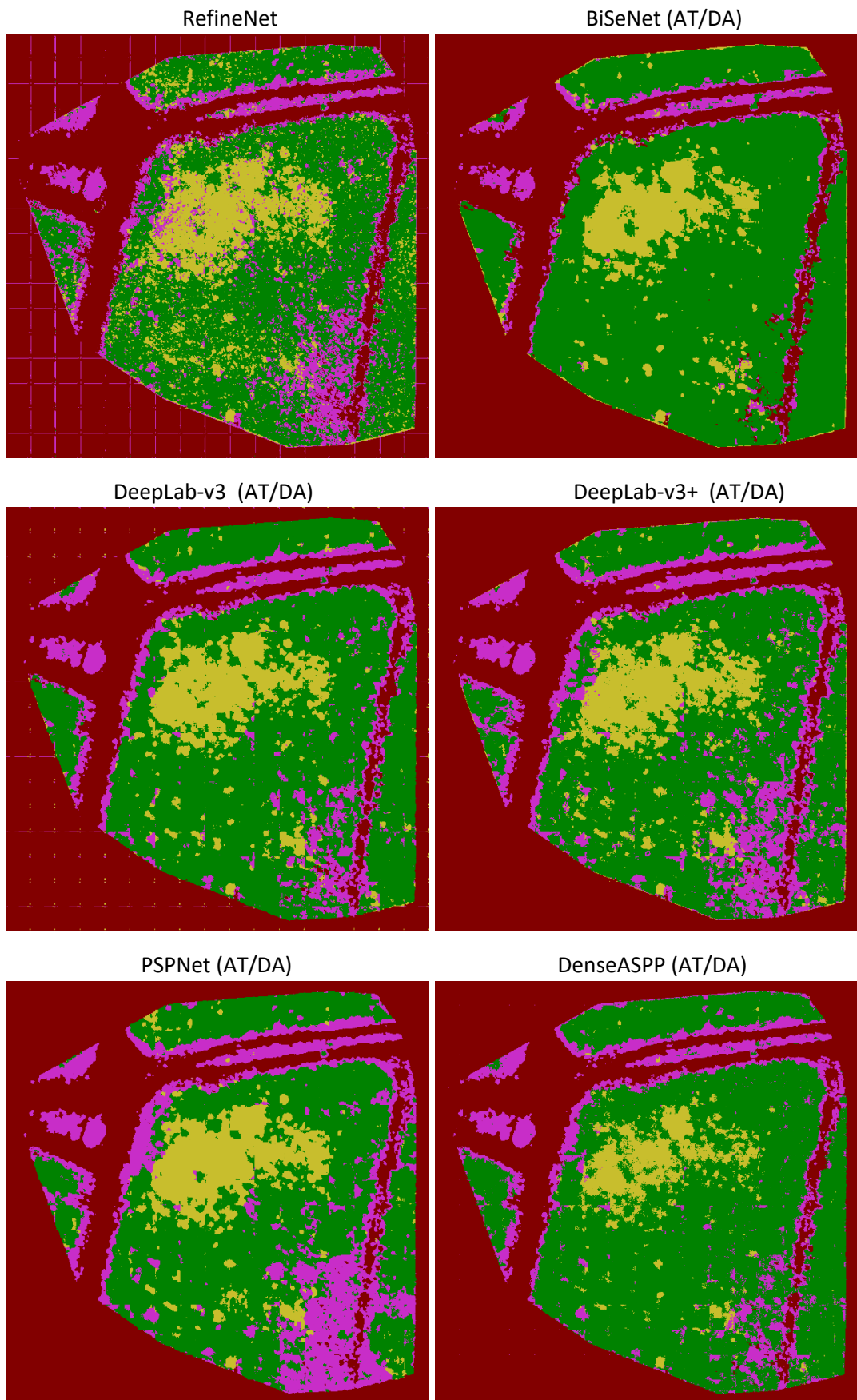


Na [Figura 5.51](#), o treinamento e teste foram realizados com o mesmo GSD e obtiveram resultados parecidos. Na [Figura 5.53](#), as redes multiescala BiSeNet e DeepLab-v3 – treinadas com blocos de treinamento  $576 \times 256$  ( $2,25 \times$  GSD) da imagem aérea e testadas com blocos de teste  $512 \times 512$  ( $1 \times$  GSD) do mosaico RGB – tiveram um resultado qualitativo superior em relação aos modelos em escala única. No entanto, a maior resolução do mosaico de teste introduz ruído na predição.

Treinamento:  $17 \times 17$  blocos de treinamento  $576 \times 256$  ( $2,25 \times$  GSD) da imagem aérea RGB.  
Entrada para predição:  $17 \times 17$  blocos de teste  $512 \times 512$  ( $1 \times$  GSD) do mosaico RGB.  
Saída prevista:  $17 \times 17$  blocos de  $512 \times 512$  pixels ( $4.352 \times 4.352$  pixels).



**Figura 5.53** – Predições do mosaico de tamanho  $8.704 \times 8.704$  pixels, sem máscara de borda, a partir dos blocos de teste  $512 \times 512$  ( $1 \times$  GSD) do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento  $576 \times 256$  ( $2,25 \times$  GSD) das imagens aéreas RGB (continuação).

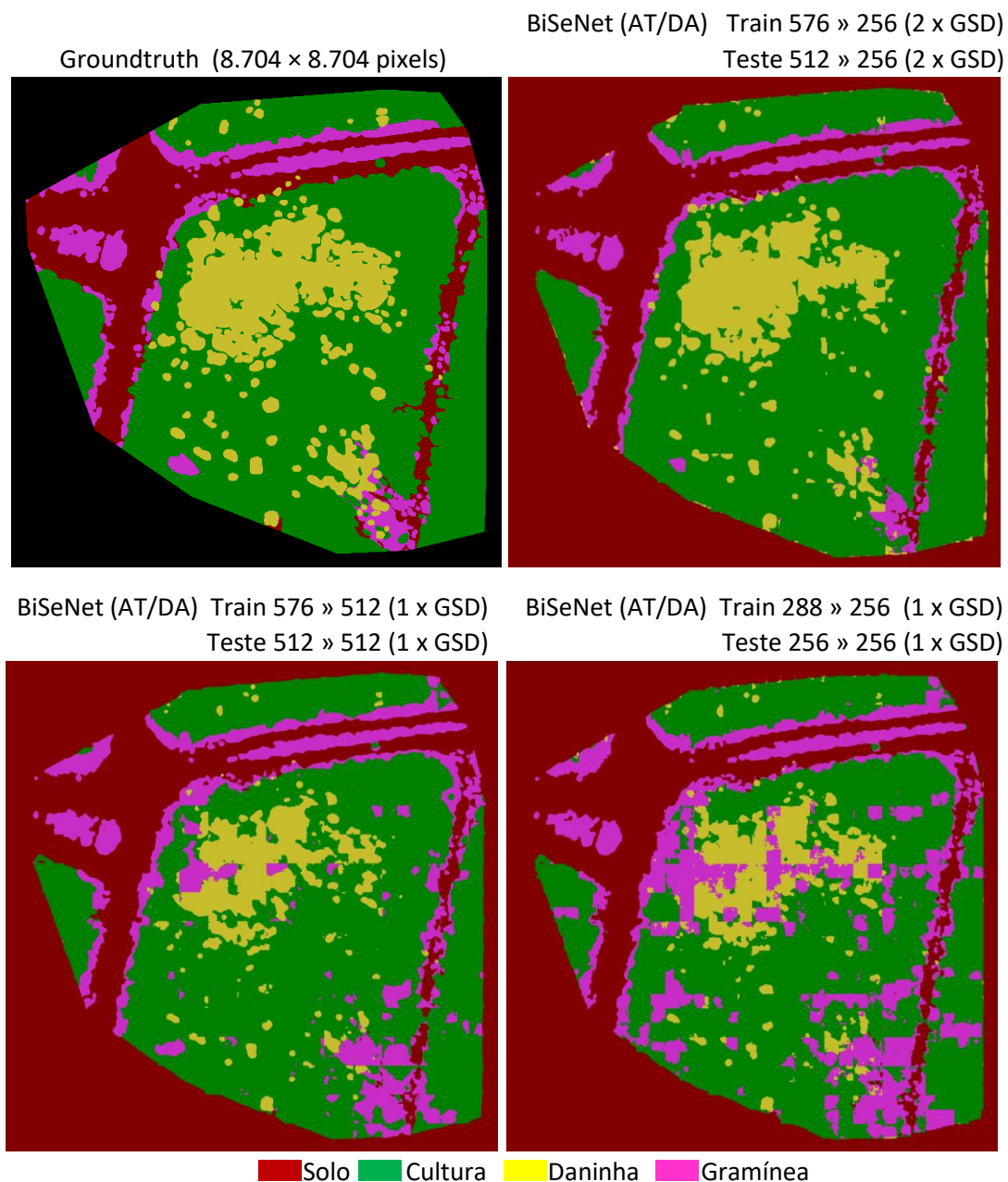


**Figura 5.53** – *Continuação da figura anterior.*

### 5.3.4.1 BiSeNet com diferentes tamanhos de bloco e GSD de treinamento

Realizamos outras análises qualitativas com a rede BiSeNet, uma vez que esta obteve desempenho superior, com menor tempo de treinamento e inferência. A [Figura 5.54](#) mostra o resultado qualitativo correspondente à [Tabela 5.11](#) (com diferentes estratégias de treinamento). A rede BiSeNet apresentou melhor qualidade visual com subamostragem na entrada da rede, ou seja, treinamento e teste com blocos de  $576 \times 256$  ( $2 \times \text{GSD}$ ). Isto permite obter informação de maior contexto semântico, uma vez que o mosaico tem alta resolução com GSD médio de  $1,16 \text{ cm/pixel}$ .

Treinamento: diferentes tamanhos de blocos e GSD da imagem aérea RGB.  
 Entrada para predição: tamanhos de blocos e GSD correspondente do mosaico RGB.  
 Saída prevista: diferentes tamanhos.



**Figura 5.54** – Predição do mosaico sem máscara de borda, a partir dos blocos de treinamento e teste do mosaico RGB com diferentes tamanhos e GSD de entrada.

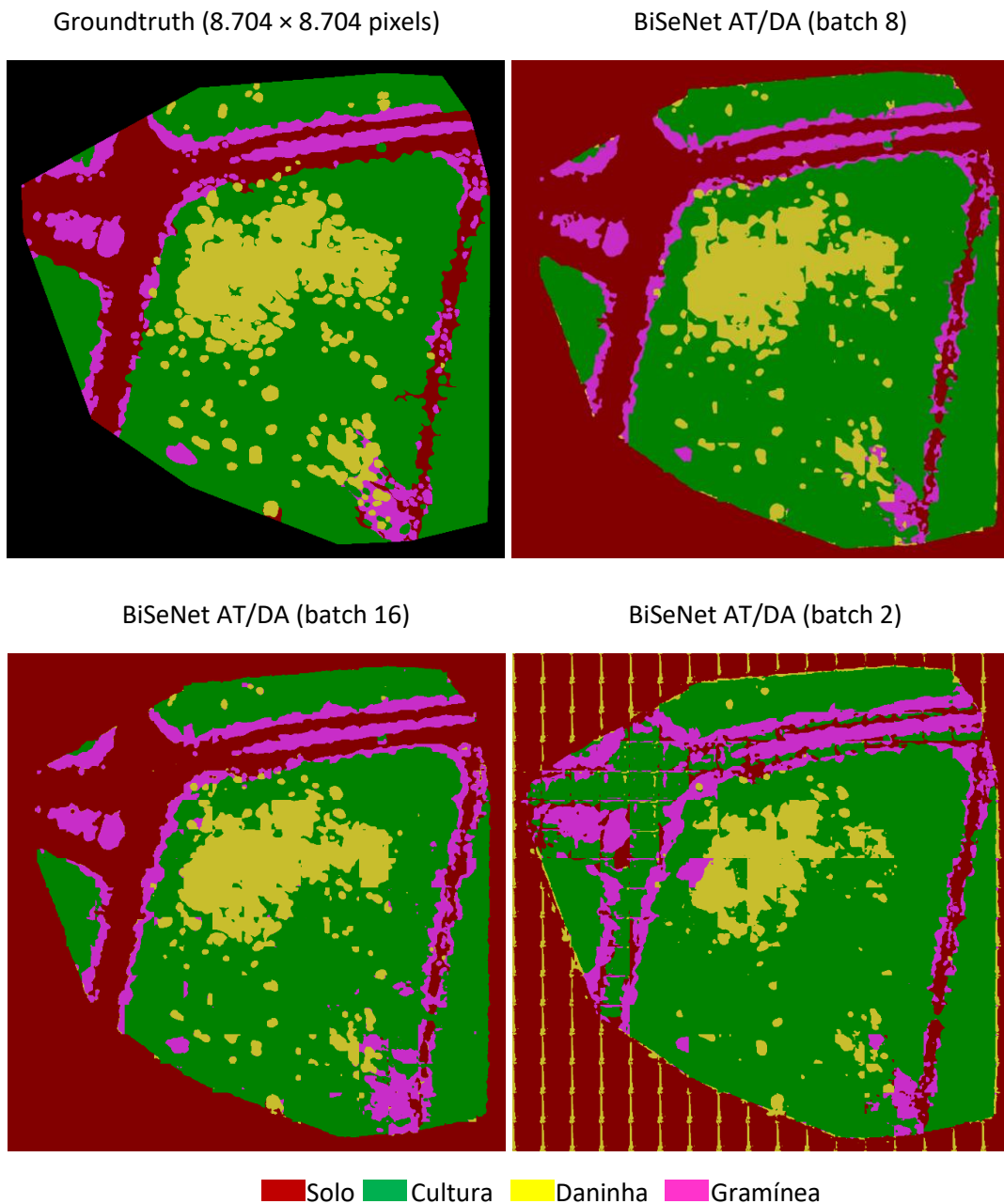
### 5.3.4.2 BiSeNet com diferentes tamanhos de *batch*

A [Figura 5.55](#) mostra o resultado qualitativo correspondente à [Tabela 5.11](#), com diferentes tamanhos de *batch*, mas mesmo tamanho de blocos e GSD de entrada. Os tamanhos de *batch* maiores apresentaram melhor resultado visual, portanto, todos os treinamentos realizados neste trabalho utilizaram  $batch = 8$ .

Treinamento:  $17 \times 17$  blocos de treinamento  $576 \gg 256$  ( $2,25 \times GSD$ ) da imagem aérea RGB.

Entrada para predição:  $17 \times 17$  blocos de teste  $512 \gg 256$  ( $2 \times GSD$ ) do mosaico RGB.

Saída prevista:  $17 \times 17$  blocos de  $256 \times 256$  pixels.



**Figura 5.55** – Predição do mosaico sem máscara de borda, a partir dos blocos de teste  $512 \gg 256$  ( $2 \times GSD$ ) do mosaico RGB, com diferentes tamanhos de *batch*.



### 5.3.4.3 BiSeNet (2 x GSD) testado com diferentes tamanhos e GSD

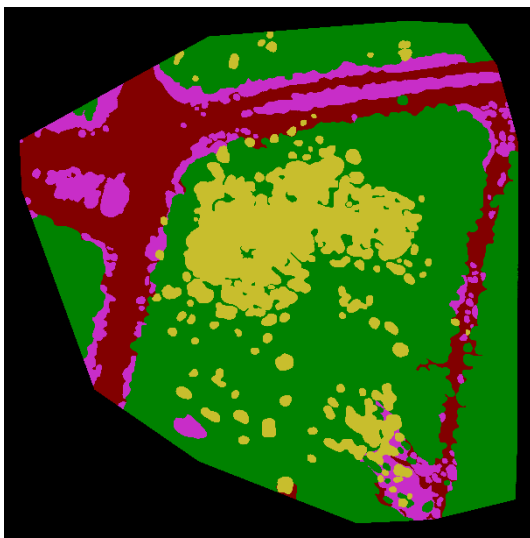
A [Figura 5.56](#) mostra o resultado qualitativo da rede BiSeNet – treinada com blocos  $576 \times 256$  ( $2,25 \times \text{GSD}$ ) das imagens aéreas RGB e testada em vários tamanhos de blocos e GSD do mosaico RGB – correspondente à linha rosa do gráfico da [Figura 5.42](#). As previsões de blocos de teste com GSD mais próximo do treinamento, ou seja, aproximadamente  $2 \times \text{GSD}$ , obtiveram melhor resultado visual.

Treinamento:  $17 \times 17$  blocos de treinamento  $576 \times 256$  ( $2,25 \times \text{GSD}$ ) da imagem aérea RGB.

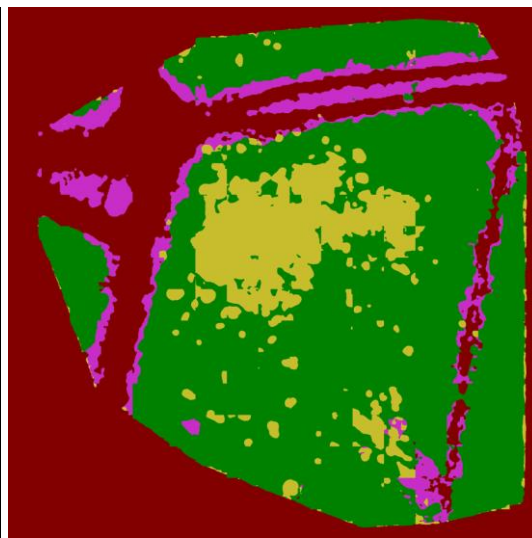
Entrada para predição: diferentes tamanhos de blocos e GSD do mosaico RGB.

Saída prevista: diferentes tamanhos.

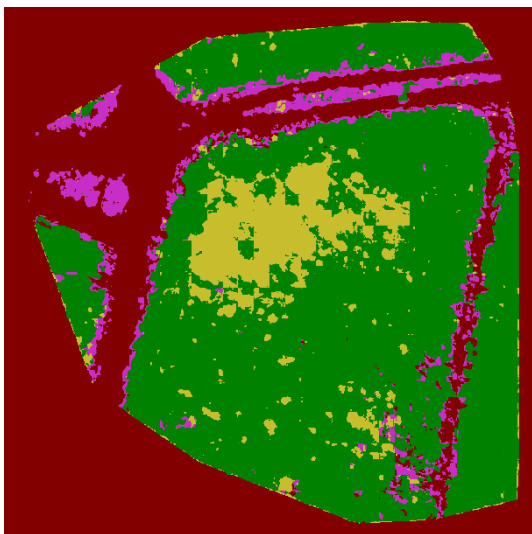
Groundtruth ( $8.704 \times 8.704$  pixels)



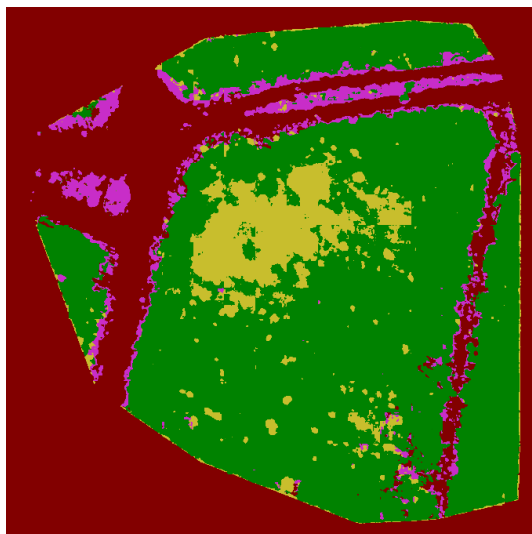
BiSeNet (AT/DA) Teste  $512 \times 256$  ( $2 \times \text{GSD}$ )



BiSeNet (AT/DA) Teste  $256 \times 256$  ( $1 \times \text{GSD}$ )



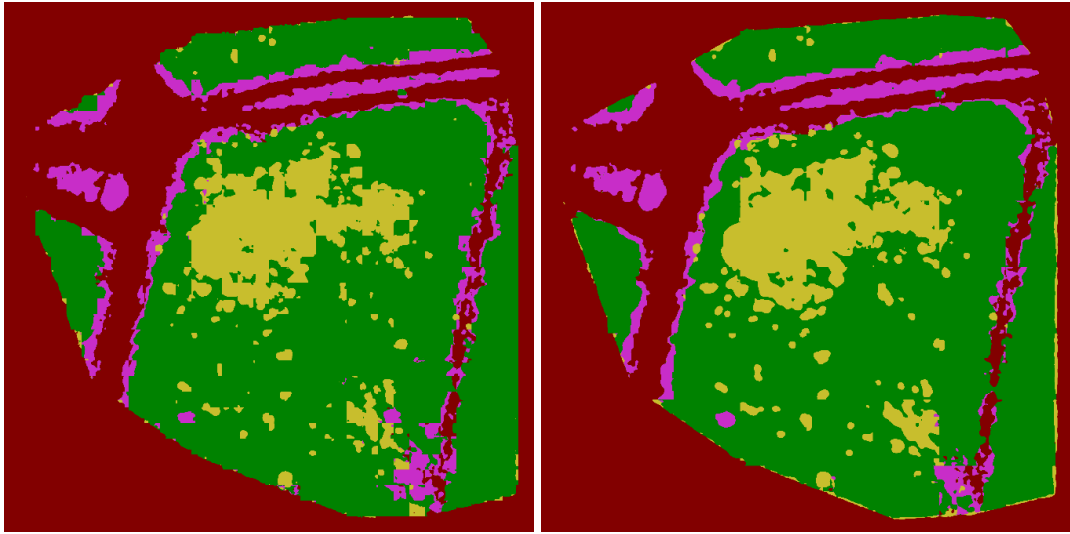
BiSeNet (AT/DA) Teste  $512 \times 512$  ( $1 \times \text{GSD}$ )



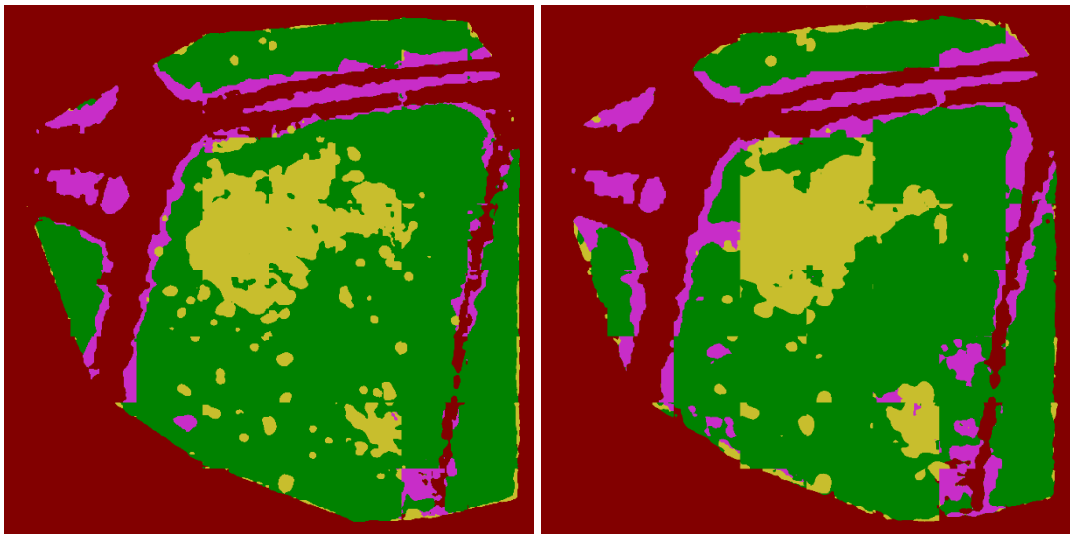
■ Solo 
 ■ Cultura 
 ■ Daninha 
 ■ Gramínea 
 (continua)



BiSeNet (AT/DA) Teste 256 » 128 (2 x GSD)    BiSeNet (AT/DA) Teste 1.088 » 512 (2,125 x GSD)



BiSeNet (AT/DA) Teste 1.088 » 384 (2,8 x GSD)    BiSeNet (AT/DA) Teste 1.088 » 256 (4,25 x GSD)



■ Solo   ■ Cultura   ■ Daninha   ■ Gramínea

**Figura 5.56** – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelos BiSeBet treinado com blocos de treinamento de tamanho 576 » 256 (2,25 x GSD) das imagens aéreas RGB.

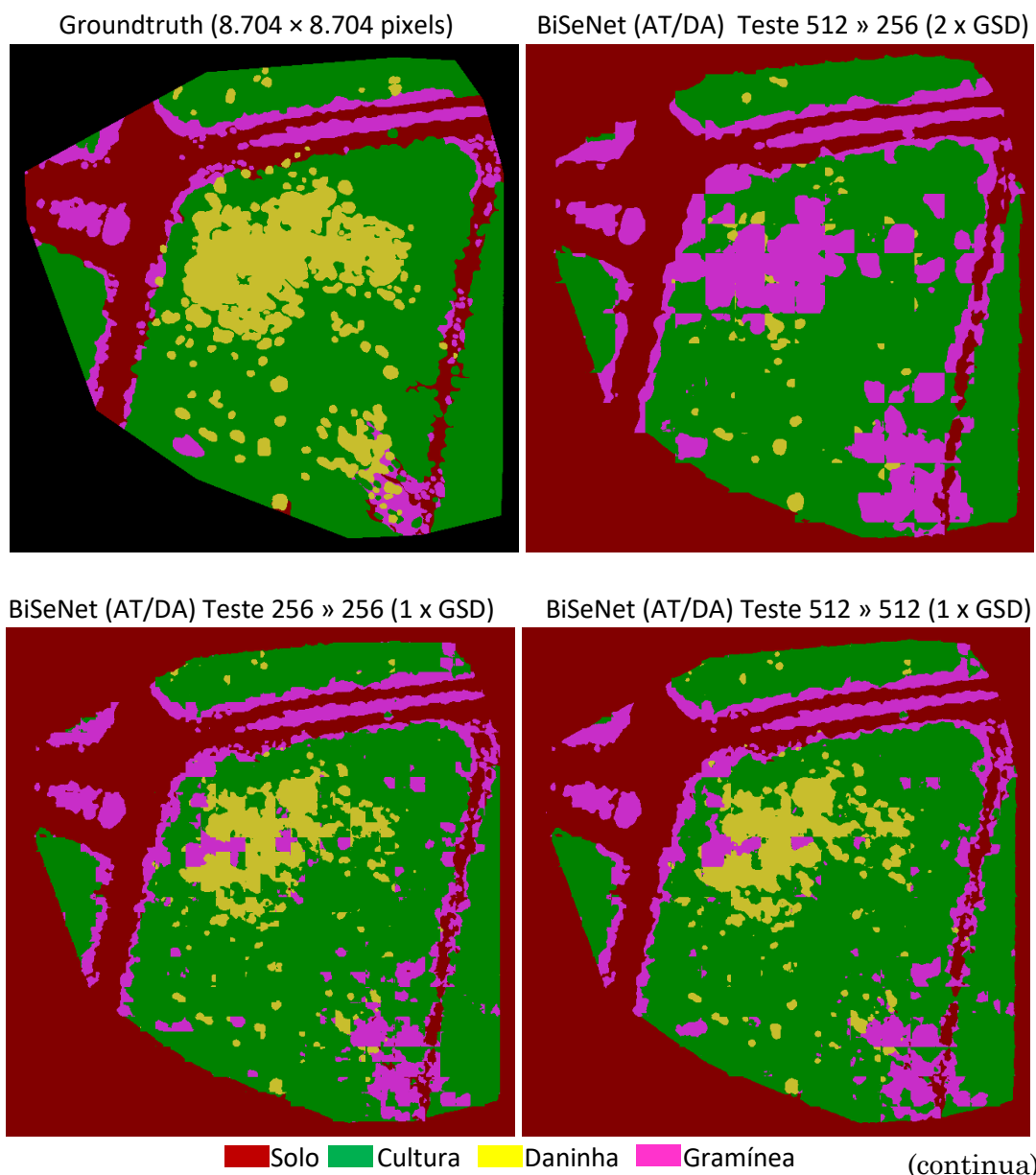
### 5.3.4.4 BiSeNet (1 x GSD) testado com diferentes tamanhos e GSD

A [Figura 5.57](#) mostra o resultado qualitativo da rede BiSeNet – treinada com blocos 576 » 512 (1,125 x GSD) das imagens aéreas RGB e testada em vários tamanhos de blocos e GSD do mosaico RGB – correspondente à linha verde do gráfico da [Figura 5.42](#). As previsões com GSD mais próximo do treinamento tiveram melhor resultado visual; por exemplo, teste com blocos 512 » 512 (1 x GSD) ou 256 » 256 (1 x GSD). Entretanto, como o mosaico tem alta resolução espacial, treinar com resolução de 1 x GSD gera um menor contexto semântico na saída do *backbone* da rede BiSeNet e, conseqüentemente, diminuição significativa de desempenho em relação à [Figura 5.56](#). Com um campo de visão (FoV) menor, cobrindo poucos *pixels* de contexto da imagem de entrada de alta resolução, as classes de vegetação – cana, daninha e gramínea – podem ser facilmente confundidas.

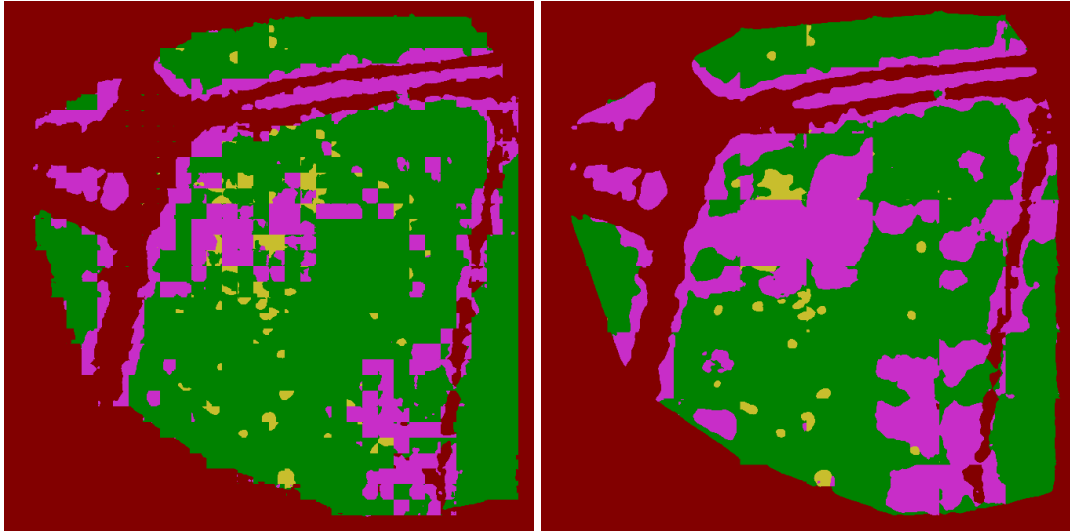
Treinamento: 17 × 17 blocos de treinamento 576 » 512 (1,125 x GSD) da imagem aérea RGB.

Entrada para predição: diferentes tamanhos de blocos e GSD do mosaico RGB.

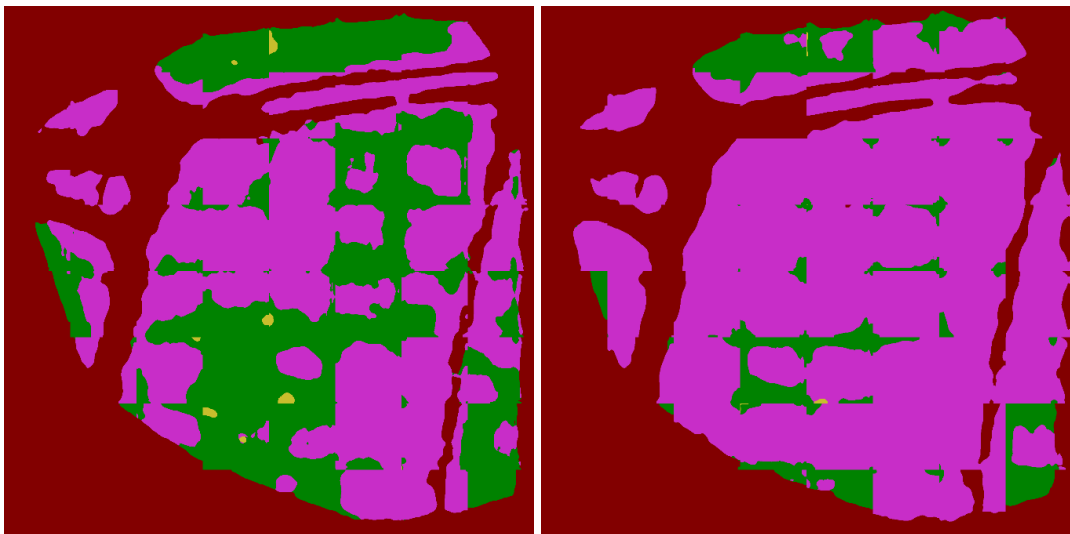
Saída prevista: diferentes tamanhos.



BiSeNet (AT/DA) Teste 256 » 128 (2 x GSD)    BiSeNet (AT/DA) Teste 1.088 » 512 (2,125 x GSD)



BiSeNet (AT/DA) Teste 1.088 » 384 (2,8 x GSD)    BiSeNet (AT/DA) Teste 1.088 » 256 (4,25 x GSD)



■ Solo   ■ Cultura   ■ Daninha   ■ Gramínea

**Figura 5.57** – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB.

### 5.3.5 Análise de requisitos para segmentação semântica na agricultura

Duas condições principais devem ser levadas em consideração para realizar um mapeamento aéreo de VANT, com o objetivo de obter um mapa da distribuição de plantas daninhas em uma cultura agrícola: (1) captura de imagens com uma resolução espacial suficiente para garantir a discriminação de ervas daninhas; e (2) cobertura da maior área possível, com maior altura de voo, para diminuir a duração de voo do VANT e o número de imagens necessárias para processamento do ortomosaico [PEÑA *et al.* 2015].

A resolução espacial da imagem (tamanho do *pixel* ou GSD em *cm/pixel*) e a área de cobertura no solo são afetadas pela altura de voo e sensor da câmera, como discutido na [Seção 4.2.2 do Capítulo 4](#). Quanto maior a altura de voo, menor a resolução espacial (com maior GSD) e maior a área capturada no solo. Assim, se o objetivo for a discriminação de plantas daninhas individuais nos estágios iniciais de crescimento, o tamanho do *pixel* deve ser de poucos centímetros, obtido com menor altura de voo. Porém, se o objetivo for a detecção de agrupamentos de ervas daninhas, o GSD pode ser até maior que 5 *cm/pixel*, o que corresponde a uma altura mais elevada de voo.

Nas imagens aéreas RGB, a precisão da classificação da vegetação no início da fase de crescimento pode diminuir em alturas de voo maiores devido ao tamanho reduzido das ervas daninhas e plantas cultivadas. Em outras palavras, a menor resolução espacial das imagens de plantas daninhas pequenas (que ocupam poucos *pixels* na imagem) pode causar erros de não detecção (falso negativo), subestimação de ervas daninhas ou superestimação (falso positivo) em áreas livres de daninhas. Como o tamanho do campo receptivo ou campo de visão (FoV) das redes de segmentação semântica de escala única é muito maior do que o tamanho das daninhas, elas não são corretamente detectadas, como discutido na [Seção 2.4.13 do Capítulo 2](#).

De maneira oposta, as imagens RGB de cultura agrícola na fase tardia de crescimento podem ter resultados de predição melhores em alturas de voo mais elevadas devido ao tamanho maior das plantas daninhas. Neste caso, a menor resolução espacial contribui para que o campo de visão limitado das redes de segmentação semântica de escala única tenha um tamanho suficiente para cobrir mais *pixels* e obter um maior contexto semântico para discriminar entre as plantas daninhas, cultura e gramíneas. Este efeito foi observado nos resultados visuais da rede BiSeNet, treinada com resolução espacial de 2 x GSD ([Figura 5.56](#)), que obteve melhor resultado do que o treinamento com blocos de 1 x GSD de maior resolução ([Figura 5.57](#)).

Em ambos os casos, a detecção de plantas daninhas de diferentes tamanhos e fases de crescimento – em mosaico de imagens aéreas com variações de escala durante o voo ou em mosaicos com GSDs diferentes do treinado pela rede – continua a ser um desafio para redes de segmentação semântica. Na [Seção 5.3](#) deste capítulo, mostramos que estas limitações foram parcialmente amenizadas pelas redes neurais de segmentação semântica multiescala selecionadas (em comparação com redes de escala única que tiveram os piores desempenhos). Com isso, é possível aumentar o desempenho de predição, sem necessidade de treinamento com um conjunto maior de imagens em escalas variadas; seja por aumento artificial de dados, ou por outras estratégias de treinamento multiescala.

No próximo capítulo, descrevemos uma nova arquitetura de segmentação semântica multiescala para atender aos requisitos das aplicações de agricultura de precisão usando VANTs como plataforma de mapeamento aéreo.





# Capítulo 6

## Modelos de segmentação multiescala

Neste capítulo, desenvolvemos uma nova arquitetura baseada em rede neural convolucional para segmentação semântica multiescala de mosaico de imagens aéreas de alta resolução de VANT de uma cultura de cana-de-açúcar com plantas daninhas.

### 6.1 Introdução

Todos os modelos de redes de segmentação semântica de imagens, descritos no [Capítulo 2](#) e testados no [Capítulo 5](#), utilizam uma CNN como *backbone*. A CNN reduz gradualmente a dimensão e resolução espacial dos mapas de atributos para que informações semânticas de longo alcance e maior contexto sejam capturadas na saída. Sendo assim, o modelo de segmentação deve recuperar gradualmente as informações espaciais de localização e os detalhes do objeto, aprendendo por meio de *upsampling* e de uma série de convoluções a mapear os mapas de atributos de baixa resolução (da camada mais profunda da CNN) para mapas de atributos de alta resolução (com rótulos semânticos de previsão de classe) com as mesmas dimensões da imagem de entrada. Além disso, para lidar com a dificuldade de segmentação de objetos de diferentes tamanhos, os mapas de atributos devem ser capazes de cobrir múltiplas escalas de campos receptivos.

Os modelos de segmentação semântica são divididos em categorias de acordo com a técnica utilizada para atender a estes requisitos, tais como: conexão de salto, codificador-decodificador, convolução *atrous* em série e convolução *atrous* em paralelo (ASPP), conforme discutido na [Seção 2.4.12](#).

Tanto as arquiteturas de salto quanto as arquiteturas codificador-decodificador buscam recuperar a resolução espacial perdida na CNN-base, no entanto, não abordam o problema de objetos multiescala. Isso pode ser observado no gráfico da [Figura 5.37](#), onde as redes FCN [[LONG et al. 2015](#)], SegNet [[BADRINARAYANAN et al. 2017](#)], U-Net [[RONNEBERGER et al. 2015](#)] e RefineNet [[LIN, MILAN et al. 2017](#)] obtiveram os piores resultados em escalas diferentes da originalmente treinada pela rede.

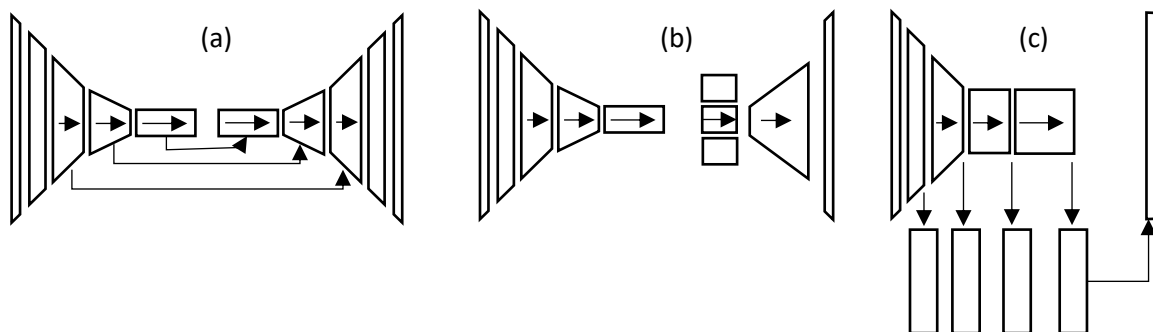
As arquiteturas que usam um módulo de contexto no topo da CNN removem as operações de subamostragem dos últimos blocos da CNN-base, com a finalidade de preservar o tamanho espacial do mapa de atributos. Simultaneamente, o último bloco da CNN emprega a convolução *atrous* para ampliar o campo de visão dos filtros. Porém, apesar do módulo extra usar várias convoluções *atrous* em série (com diferentes taxas de dilatação) para capturar gradualmente informações de contexto de longo alcance (com maior campo receptivo e campo de visão), ainda assim, ele será incapaz de agregar o contexto multiescala, uma vez que o campo receptivo final da rede tem um tamanho único.

Para contornar este problema e motivados pela pirâmide espacial de *pooling* da rede PSPNet [ZHAO *et al.* 2017], os modelos DeepLab-v2 [CHEN *et al.* 2018a] e DeepLab-v3 [CHEN *et al.* 2017] usam um módulo ASPP com camadas de convoluções *trous* paralelas de diferentes taxas de amostragem e diferentes campos de visão, capturando contexto em várias escalas. DeepLab-v3 adiciona ao ASPP um atributo global de nível de imagem (GAP), que codifica o contexto global e aumenta ainda mais o desempenho. Os mapas de atributos extraídos em cada ramo ASPP são fundidos (somados ou concatenados) para gerar o mapa de atributos multiescala final. Para melhorar a resolução espacial, DeepLab-v3+ [CHEN *et al.* 2018b] utiliza também uma conexão de salto, embora o desempenho em imagens de alta resolução (menor GSD) tenha apresentado uma queda em relação à rede DeepLab-v3, no gráfico da Figura 5.37. Contudo, uma característica de todas as versões de DeepLab é que a resolução em escala é esparsa, cobrindo o contexto em poucas escalas.

A rede DenseASPP [YANG *et al.* 2018] usa um módulo extra no topo da CNN para capturar atributos multiescala de forma densa, utilizando convoluções *trous* em série e em paralelo. Assim, a rede gera uma pirâmide de atributos muito mais densa e com maior campo receptivo. No entanto, o desempenho da substituição do ASPP pelo módulo denso com imagens de alta resolução foi inferior ao DeepLab-v3 (Figura 5.38).

Nos testes do Capítulo 5, a rede BiSeNet apresentou o melhor desempenho multiescala em nosso conjunto de dados, mas ainda tem uma perda significativa de desempenho quanto mais distante for o GSD da imagem a ser predita em relação ao GSD treinado pela rede (Figura 5.42).

Observamos que em todas as redes descritas no Capítulo 2 e avaliadas no Capítulo 5, o fluxo de processamento para reconstrução da alta resolução dos mapas de atributos das camadas iniciais da rede ocorre a partir do topo da CNN (com representações de atributos mais abstratos de baixa resolução), passa por camadas intermediárias com operações intercaladas de sobreamostragem e deconvolução, seguindo até a camada final da rede de segmentação com a mesma resolução da entrada. Os modelos de rede com refinamento sucessivo de mapas de atributos – com conexão de salto ou codificador-decodificador (Figura 6.1a), como FCN, SegNet, U-Net e RefineNet, seguem o mesmo padrão, ou seja, a saída da CNN de baixa resolução serve como entrada do bloco seguinte, que recupera os mapas de atributos com maior resolução e dimensão espacial, seguindo um fluxo sequencial de informação de baixo para cima. Os modelos PSPNet, DeepLab-v3 e DenseASPP utilizam um módulo de pirâmide após a camada superior da CNN para obter informações de contexto em diferentes escalas, seguindo o mesmo fluxo de informação dos mapas de atributos da menor para maior resolução (Figura 6.1b).



**Figura 6.1** – Fluxo de informação dos mapas de atributos nos blocos convolucionais. A altura representa dimensão espacial e a largura representa a quantidade de canais nos mapas de atributos.

A [Figura 6.1c](#) mostra outro esquema, onde o fluxo de informação dos mapas de atributos ocorre de forma paralela; ou seja, os mapas de atributos de alta a baixa resolução da CNN são sobreamostrados com diferentes taxas para obter a mesma dimensão espacial, antes de serem fundidos por adição ou concatenação, para incorporar atributos de diferentes escalas. O mapa multiescala resultante é, então, sobreamostrado novamente para recuperar a mesma dimensão da entrada. Desta forma, informações de detalhes dos objetos nas camadas inferiores de maior resolução da CNN são preservadas e anexadas diretamente a informações de maior contexto semântico das camadas superiores de baixa resolução. As redes Hypercolumns [[HARIHARAN et al. 2014a](#)], DeepLab-MsC (v1) [[CHEN et al. 2015](#)] e, de forma diferenciada, a rede BiSeNet [[YU et al. 2018](#)] com dois caminhos, seguem este padrão, concatenando mapas de atributos das camadas intermediárias da CNN, como mostrado na [Figura 2.58](#), [Figura 2.92](#) e [Figura 2.118](#), respectivamente.

Com o objetivo de atender aos requisitos das aplicações de segmentação semântica de plantas daninhas em imagens aéreas agrícolas de alta resolução de VANT com variações de GSD, neste capítulo desenvolvemos uma série de modelos de rede multiescala. Utilizamos a arquitetura com fluxo paralelo de mapas de atributos ([Figura 6.1c](#)) nas saídas dos quatro últimos blocos convolucionais da CNN, com intuito de preservar as informações de maior resolução em diferentes escalas e, simultaneamente, obter informação de longo alcance e maior contexto semântico. Além disso, introduzimos novos recursos, baseados nas características estruturais das redes de segmentação apresentadas na [Seção 2.4](#) do [Capítulo 2](#) e avaliadas em nosso conjunto de dados no [Capítulo 5](#), com o objetivo de obter atributos multiescala densa de maior campo de visão para imagens de alta resolução. Assim, iniciamos este capítulo com a descrição de um modelo de referência (*baseline*) e, a partir dele, descrevemos outros modelos aprimorados com melhor desempenho multiescala. Em seguida, avaliamos e comparamos o desempenho quantitativo dos modelos desenvolvidos. Finalmente, selecionamos e descrevemos o modelo proposto que obteve melhor desempenho multiescala.

## 6.2 Descrição dos modelos

### 6.2.1 Modelo de referência

Primeiro, desenvolvemos um modelo de referência (denominado modelo A1) utilizando um *backbone* VGG-16 pré-treinado no ImageNet, como mostra o [Figura 6.2](#). Os dois últimos blocos de convolução do *backbone*, Conv4 e Conv5, foram redefinidos; isto é, as operações de subamostragem foram removidas para manter uma maior resolução na saída do *backbone* (1/8 da entrada em vez de 1/32) e a convolução *atrous* com taxa de dilatação  $r = 2$  foi usada no último bloco para aumentar o campo de visão do filtro.

Para extrair atributos dos quatro blocos superiores com a mesma dimensão espacial que possibilite as operações de adição ou concatenação, primeiro um filtro de convolução  $3 \times 3$  (após o preenchimento de zeros) reduz o número de canais dos mapas de atributos dos blocos 4 e 5 de 512 para 256 canais. Os mapas de atributos provenientes dos blocos 2, 3, 4 e 5 são, então, sobreamostrados com interpolação bilinear de 2x (bloco 2) e 4x (blocos 3, 4 e 5) para obter um tamanho espacial de 1/2 da entrada. Em cada ramo sobreamostrado de 4x, duas operações de convolução  $3 \times 3$  densificam os atributos sobreamostrados e reduzem gradualmente o número de canais de 256 para 128. Por exemplo, com uma imagem de entrada de  $256 \times 256 \times 3$ , agora temos mapas de atributos com o mesmo tamanho espacial de  $128 \times 128 \times 128$  nos quatro ramos.

Em seguida, os mapas de atributos dos quatro ramos paralelos são adicionados (ou concatenados) agregando informações com diferentes escalas de contexto. Para obter a resolução espacial original da entrada, o mapa agregado com fator de 2 é sobreamostrado com interpolação bilinear de 2x. Em seguida, dois filtros de convolução  $3 \times 3$  reduzem gradualmente o número de canais de 128 (ou  $4 \times 128$  no caso de concatenação) para 64. O filtro de convolução  $1 \times 1$  gera os mapas de predição, reduzindo o número de canais para  $C = 4$  classes. Por fim, uma camada *softmax* gera o mapa de probabilidade de classes.

Esta arquitetura permite utilizar diretamente a informação de várias resoluções nas saídas dos blocos convolucionais do *backbone*, para gerar mapas de previsão com melhor resolução nos limites dos objetos. Desta forma, em contraste com as arquiteturas de salto, os mapas de atributos de baixa resolução não são progressivamente sobreamostrados e refinados com mapas de atributos de maior resolução de blocos anteriores do *backbone*. Diferentemente das arquiteturas codificador-decodificador, que recuperam progressivamente a resolução por deconvolução dos mapas de atributos de mais baixa resolução, esta arquitetura utiliza diretamente os mapas de atributos de diferentes contextos, agregados em cada *pixel*, de forma semelhante a rede Hypercolumns.

A principal diferença entre esta rede de referência (Figura 6.2) com Hypercolumns (Figura 2.58) e DeepLab-MsC (Figura 2.92) é o fato dela ser totalmente convolucional (sem as camadas totalmente conectadas da CNN) para obter uma melhor resolução na saída. Entretanto, o gráfico de desempenho na Figura 6.4 mostra que esta solução simples não tem bom desempenho multiescala para segmentar objetos de diferentes tamanhos. Além disso, o campo de visão é pequeno em imagens de alta resolução e, portanto, apresenta as mesmas limitações das arquiteturas de conexão de salto e codificador-decodificador (como discutido na Seção 2.4.13 e demonstrado nos resultados da Seção 5.3.1.3).

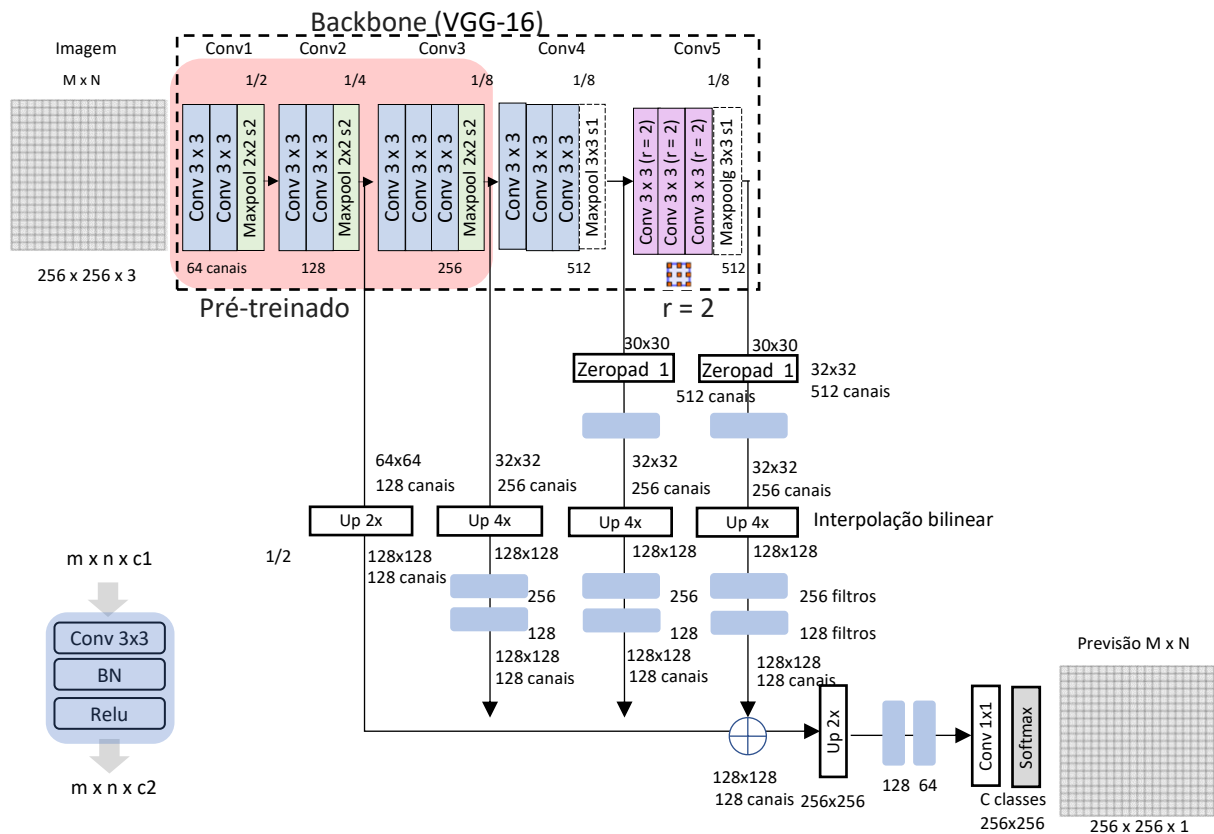
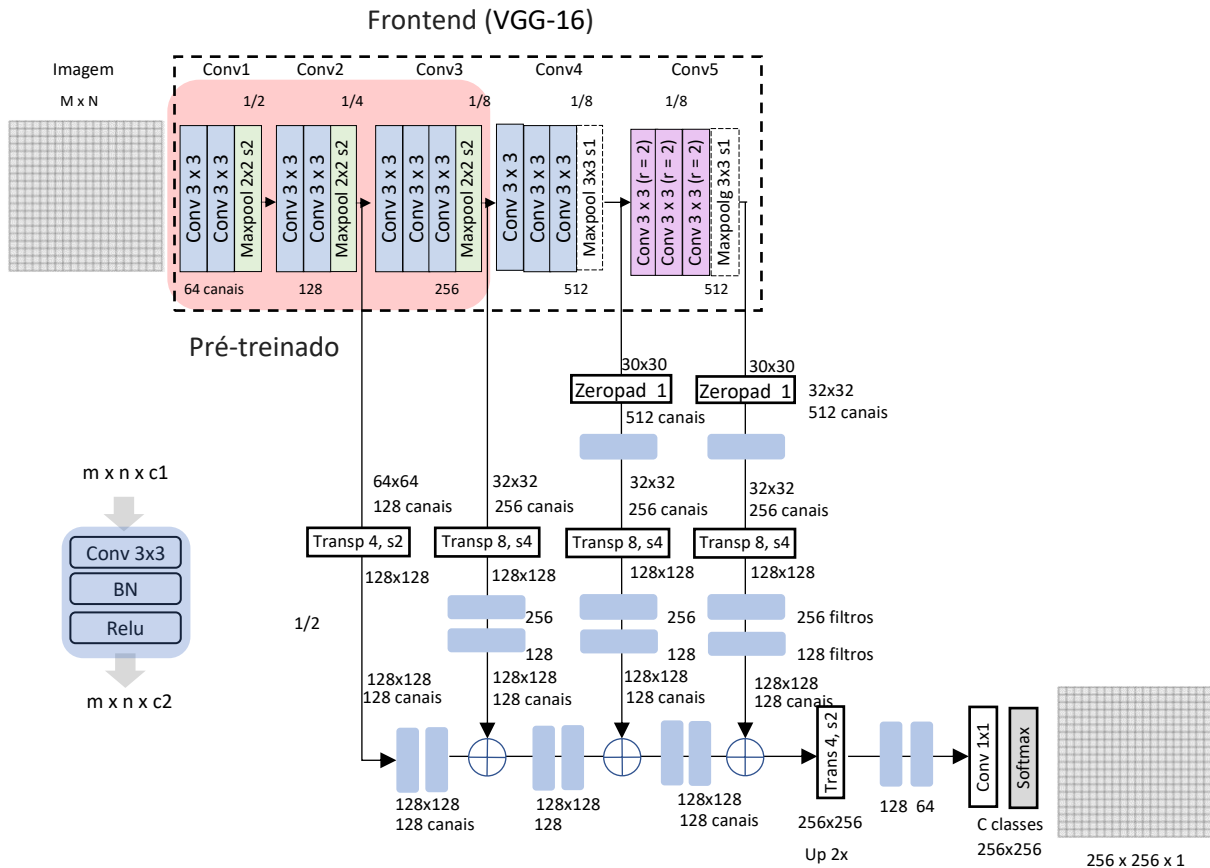
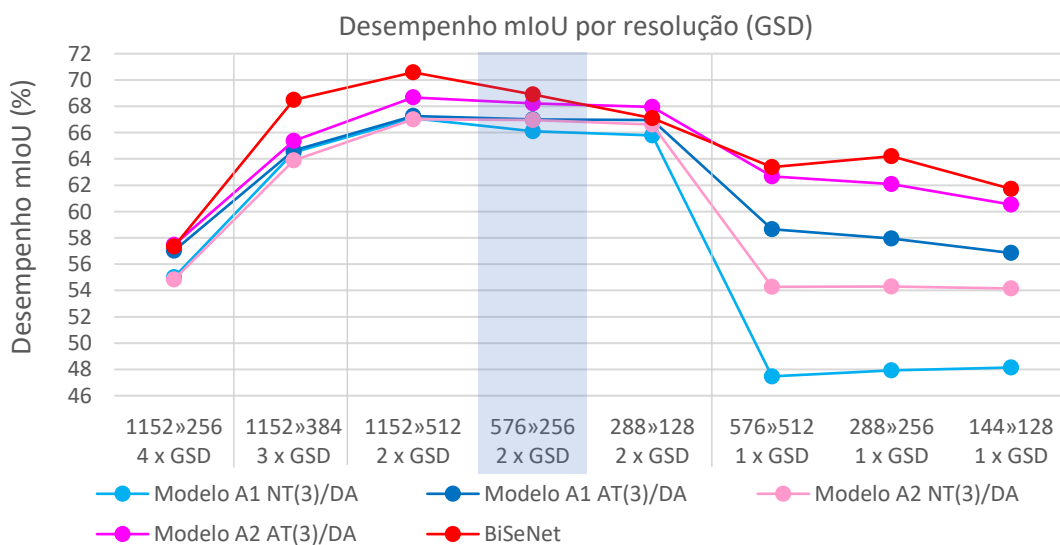


Figura 6.2 – Modelo A1: modelo de referência.

Podemos observar no gráfico da [Figura 6.4](#) que – em relação ao modelo A1 NT(3)/DA sem reajuste de pesos – ajustar os três primeiros blocos da CNN, pré-treinados no ImageNet (modelo A1 AT(3)/DA), melhora o desempenho usando imagens de entrada de alta resolução espacial (GSD menores). Para melhorar ainda mais este resultado, utilizamos a convolução transposta no modelo A2 ([Figura 6.3](#)) para fazer *upsampling* dos mapas de atributos dos quatro ramos, e também adicionamos convoluções antes de cada operação de adição para aprender os atributos. Com isso, o desempenho multiescala do modelo A2 AT(3)/DA se aproxima do modelo BiSeNet, como mostrado na [Figura 6.4](#).



**Figura 6.3** – Modelo A2: modelo de referência com atributos multiescala.



**Figura 6.4** – Comparação de desempenho dos modelos A1 e A2.



## 6.2.2 Modelo com bloco de convolução denso paralelo

No modelo B, utilizamos um bloco denso de convolução *atrous* em série e em paralelo, baseado no modelo DenseASPP [YANG *et al.* 2018]. Diferentemente de DenseASPP, que adiciona o bloco denso no final do *backbone*, utilizamos o bloco denso paralelo às camadas intermediárias da CNN pré-treinada no ImageNet.

As saídas dos quatro blocos convolucionais do *backbone* passam por uma convolução  $3 \times 3$  que gera uma saída com 256 canais cada. Depois, os mapas de atributos dos blocos 3, 4 e 5 são sobreamostrados com interpolação bilinear  $2x$ . Por exemplo, com uma imagem de entrada  $256 \times 256 \times 3$ , obtemos mapas de atributos  $64 \times 64 \times 256$  nos quatro ramos ligados ao bloco denso (fator 4). Uma dimensão espacial menor, por exemplo,  $32 \times 32 \times 256$ , reduziria o tempo de treinamento, mas o mapa deve ter um tamanho suficiente para não degradar o maior filtro de convolução *atrous* do bloco denso em uma convolução  $1 \times 1$ .

O bloco denso tem camadas de convolução *atrous* com diferentes taxas de dilatação  $r = [3, 6, 12, 18]$ , capazes de recuperar informações de contexto multiescala denso. Os mapas de atributos sobreamostrados, provenientes dos blocos 2, 3, 4 e 5 do *backbone*, foram inseridos diretamente no bloco denso para fornecer informações detalhadas em várias escalas da imagem de entrada. Na saída do bloco denso, todos os mapas de atributos são concatenados antes das operações finais de convolução. Em seguida, uma sobreamostragem de  $4x$  e uma camada *softmax* obtêm o mapa de predição com a mesma resolução da imagem original.

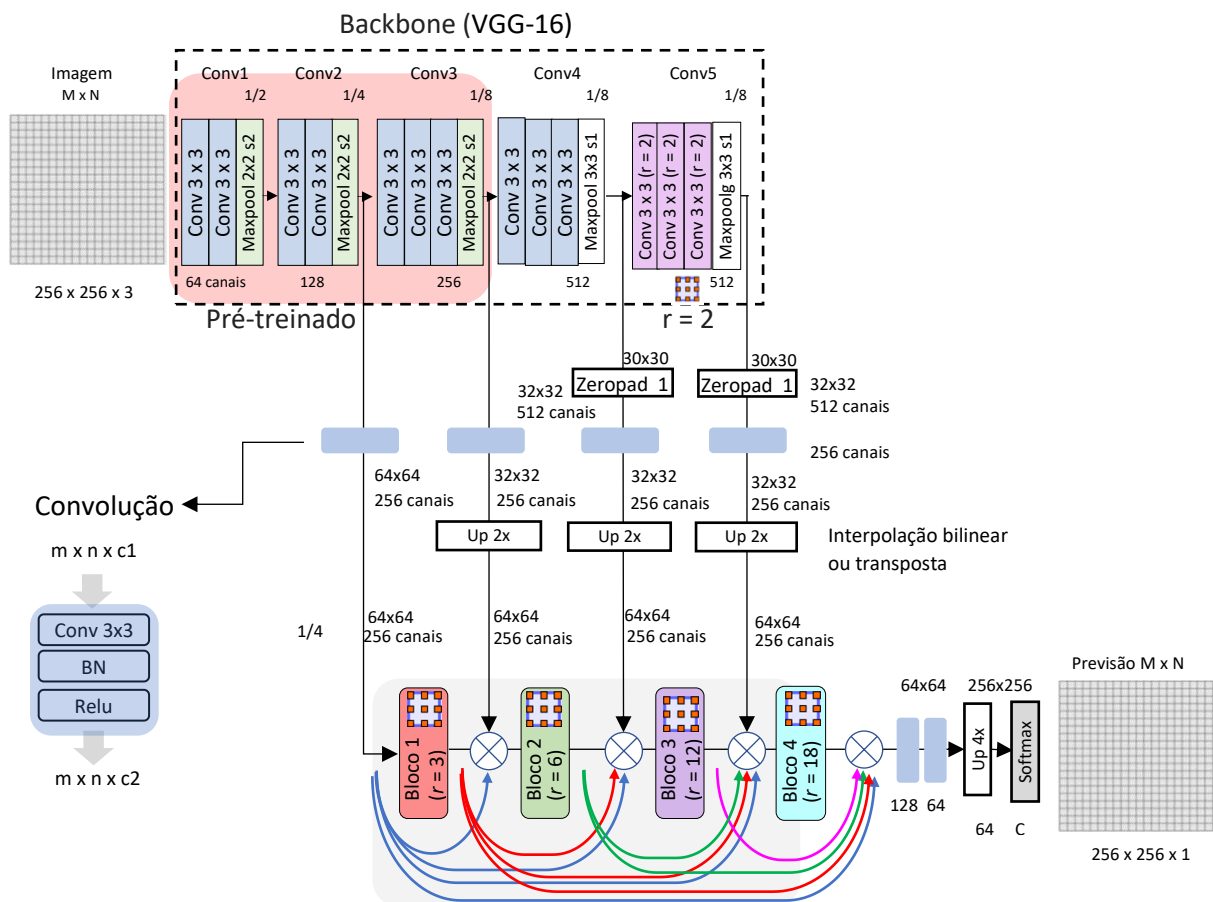
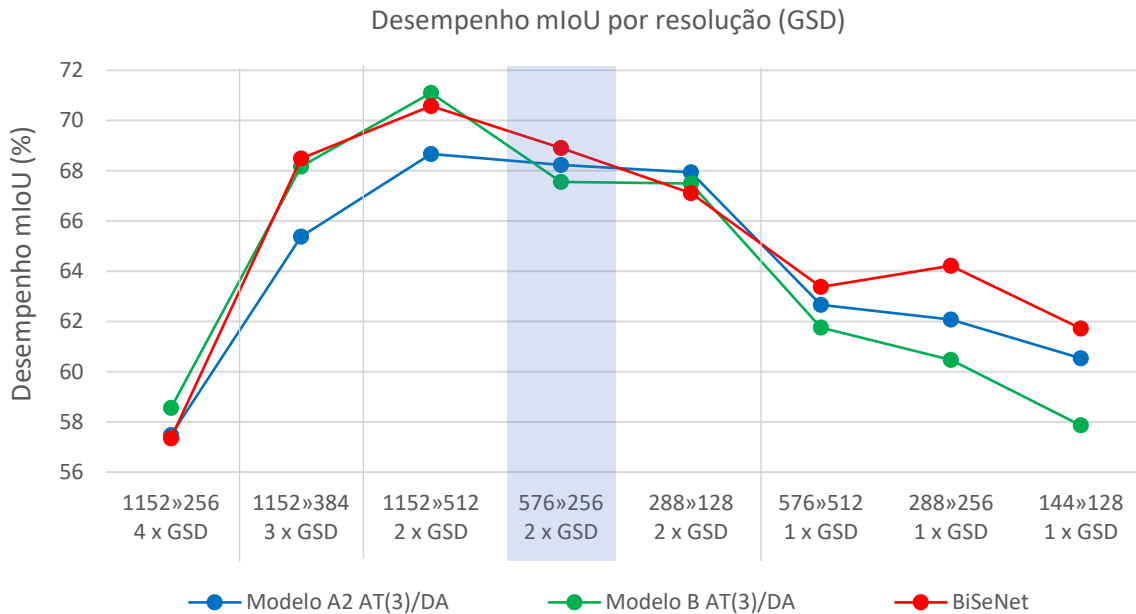


Figura 6.5 – Modelo B: Modelo com bloco de convolução denso paralelo à CNN.



**Figura 6.6** – Desempenho do modelo B com bloco de convolução denso paralelo à CNN.

No gráfico de desempenho da [Figura 6.6](#), comparamos os modelos B na configuração AT(3)/DA com o modelo A2 de referência e BiSeNet. O modelo B melhora o modelo A2 em imagens de baixa resolução (GSD maiores), mas diminui o desempenho em GSD menores, principalmente quando o bloco não tem tamanho suficiente para evitar a degeneração dos filtros *atrous* maiores em convolução  $1 \times 1$ . Por exemplo, com uma entrada de  $144 \times 128$  e fator de redução 8 na CNN, o tamanho do mapa de atributos de  $32 \times 32$  nas entradas do bloco denso degenera o filtro *atrous* com taxa  $r = 18$ , diminuindo o campo de visão em imagens de alta resolução. Portanto, blocos maiores são necessários na entrada.

### 6.2.3 Modelo com bloco denso paralelo e concatenação de atributos

Para melhorar o desempenho do modelo B, no modelo C ([Figura 6.7](#)) concatenamos os mapas de atributos intermediários do *backbone* diretamente na saída do bloco denso. Desta forma, comparando o modelo C com o modelo B na [Figura 6.8](#), a alta resolução dos mapas de atributos é recuperada com auxílio dos mapas de atributos de alta e média resolução dos blocos da CNN.

Remover o mapa de atributos de alta resolução proveniente do bloco 2 (apenas da concatenação final na [Figura 6.7](#)), degrada o desempenho em imagens de alta resolução (1 x GSD), como podemos observar comparando o desempenho do modelo “C sem B2” com o modelo C (com B2, B3, B4 e B5) no gráfico da [Figura 6.8](#). Assim, utilizar os mapas de atributos de alta resolução do bloco 2 da CNN, geralmente ignorado pelos modelos de segmentação, tem um efeito positivo no desempenho deste modelo de segmentação multiescala em imagens de todas as resoluções, quando o GSD é diferente do utilizado para treinamento da rede. No entanto, o desempenho nas imagens de baixa resolução (GSD maior) foi inferior ao modelo B anterior.

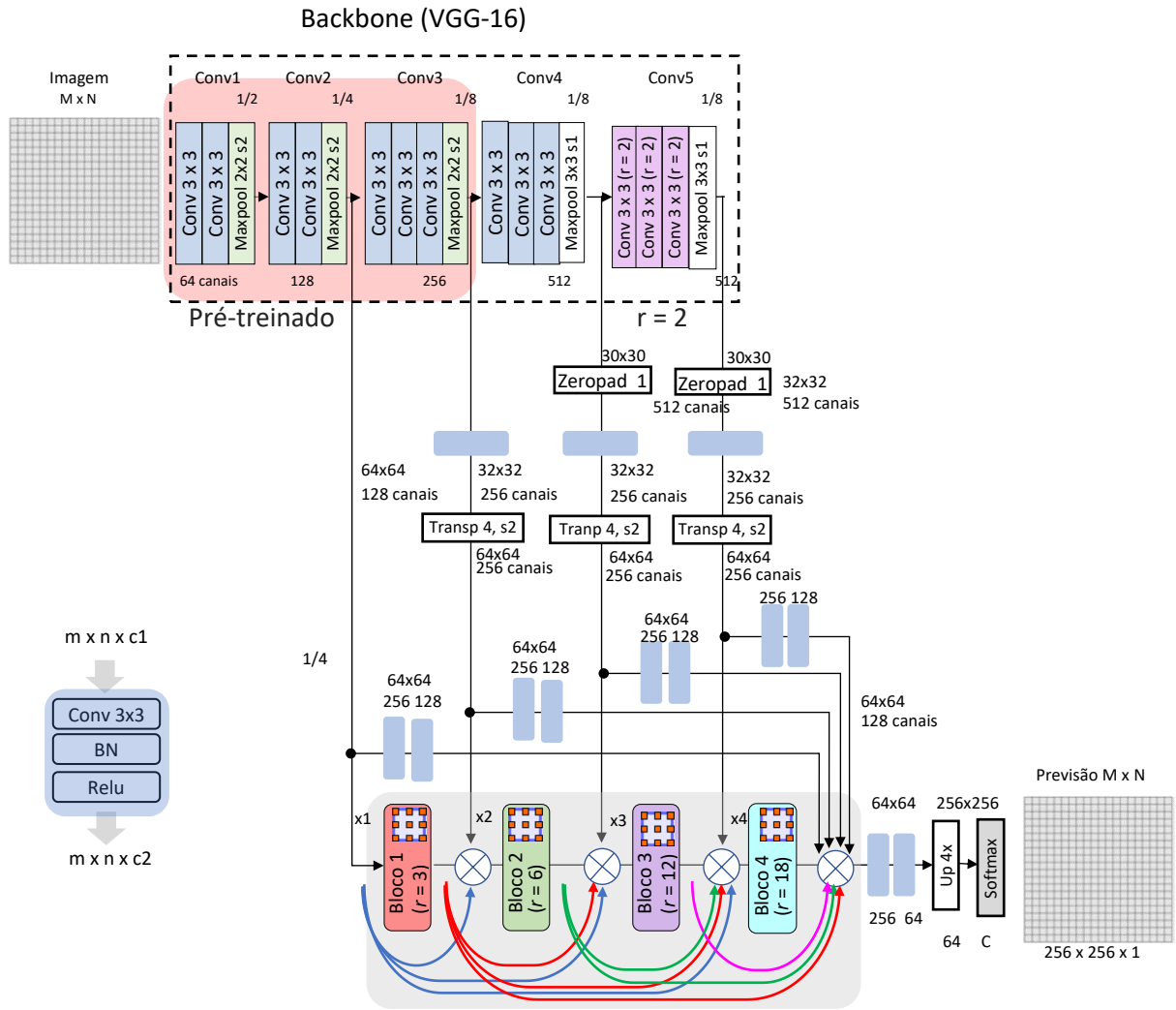


Figura 6.7 – Modelo C: Modelo com bloco de convolução denso e concatenação das camadas intermediárias da CNN.

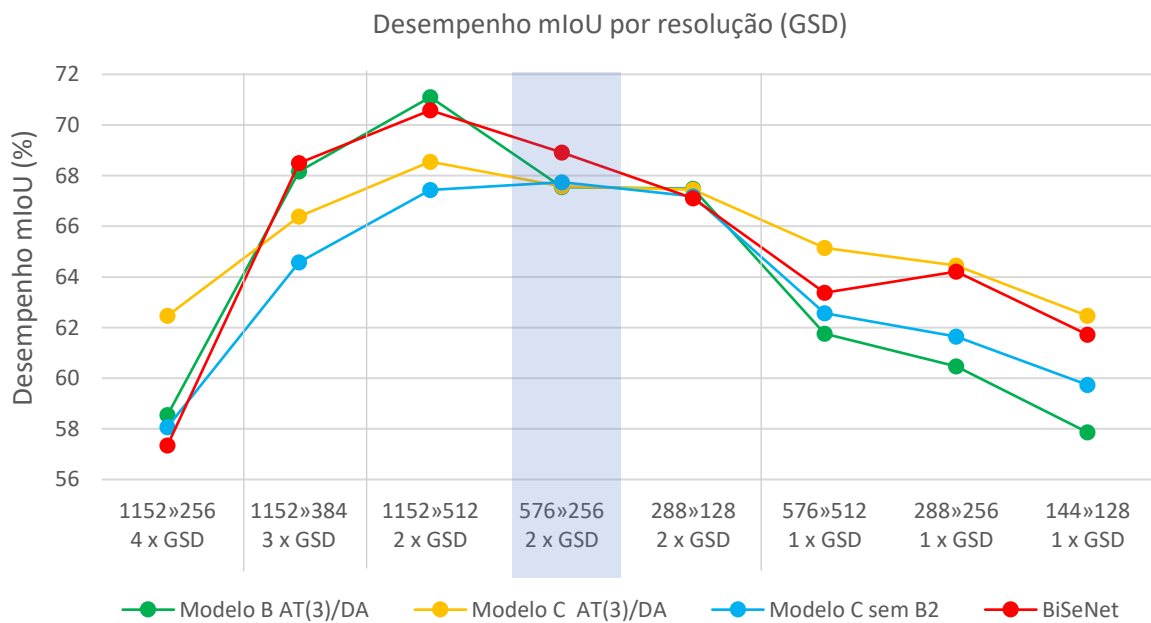


Figura 6.8 – Desempenho do modelo C com concatenação das camadas intermediárias.

### 6.2.4 Modelo com bloco denso paralelo e maior contexto

O modelo C AT(3)/DA (Figura 6.7) recupera os atributos de alta resolução em blocos com GSD menores (alta resolução espacial), mas tem desempenhos inferiores ao modelo B AT(3)/DA em blocos de baixa resolução; por exemplo, blocos de tamanho  $1152 \times 384 \text{ pixels}$  ( $4 \times \text{GSD}$ ) e  $1152 \times 512$  ( $2 \times \text{GSD}$ ) no gráfico da Figura 6.8.

Com o objetivo de aprimorar o desempenho deste modelo, no modelo D (Figura 6.9), utilizamos o *backbone* original, com subamostragem e sem convolução *atrous* nos últimos blocos da CNN (ou seja, com fator de 16 e 32 nos blocos 4 e 5, respectivamente), com o propósito de obter uma informação de longo alcance e maior contexto semântico na saída da CNN. Assim, uma sobreamostragem (no caso, convolução transposta) de 2x, 4x e 8x nos blocos 3, 4 e 5 provenientes da CNN, recupera os mapas de atributos com fator 4 nas entradas do bloco denso. Uma sobreamostragem final de 4x, antes da camada *softmax*, recupera a resolução original da entrada.

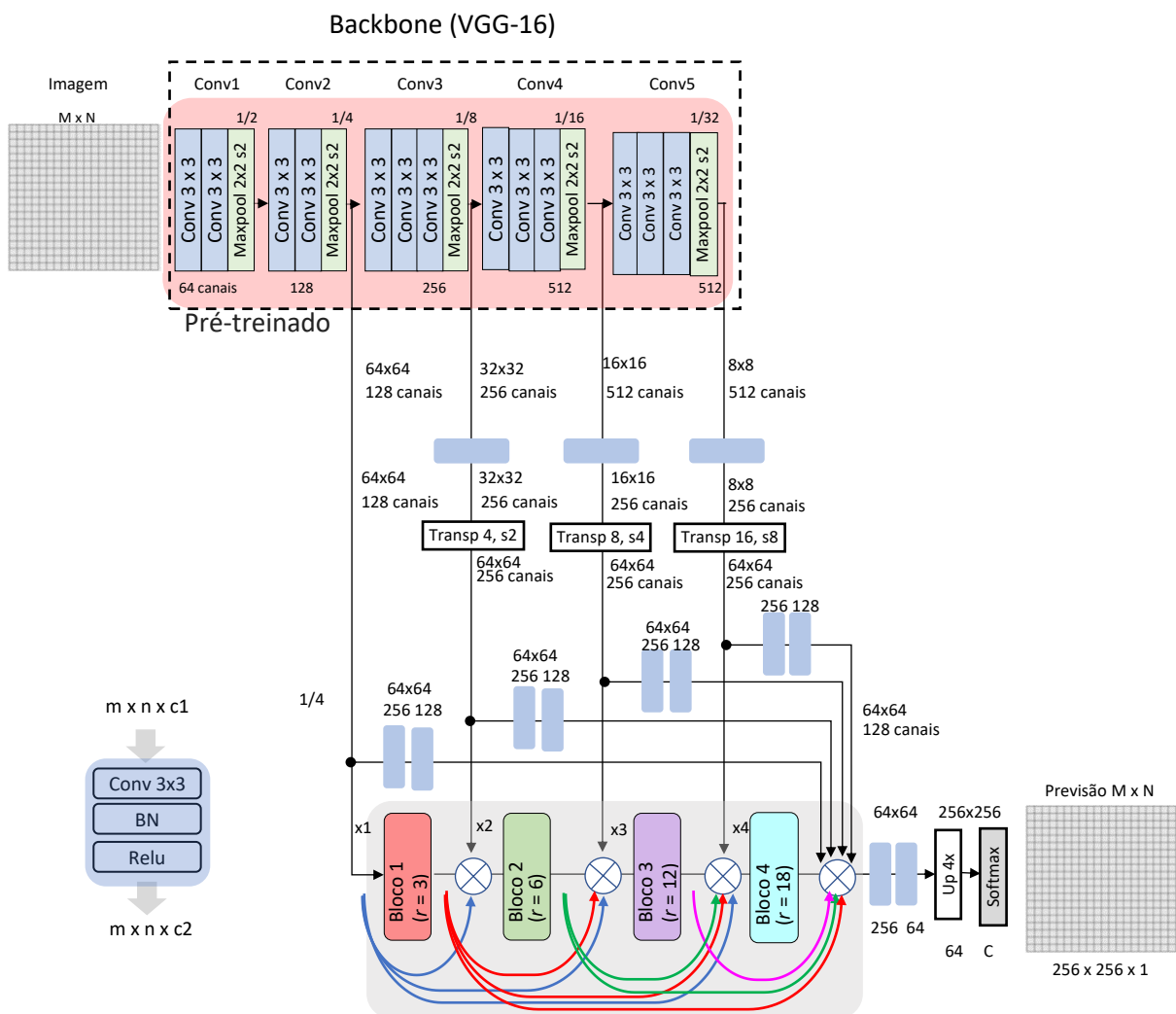
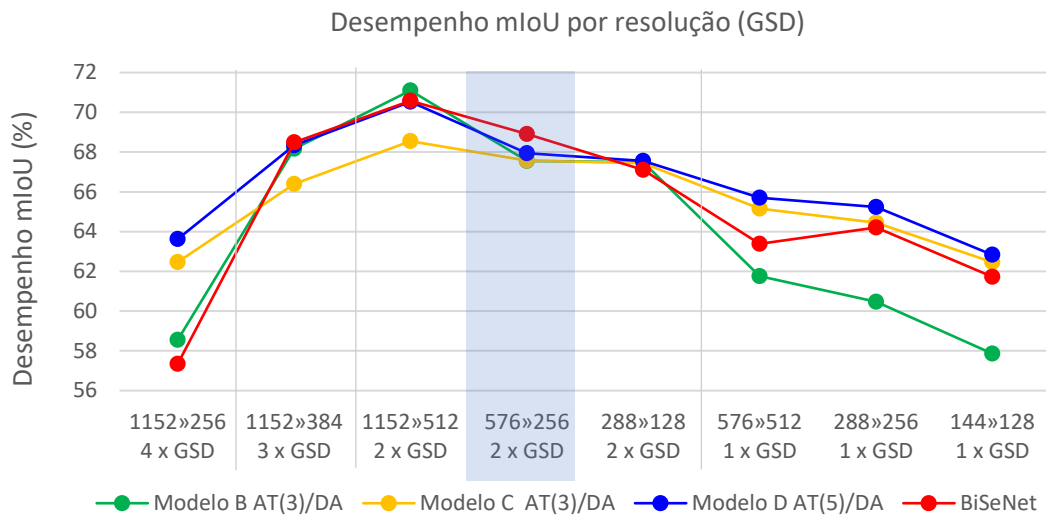


Figura 6.9 – Modelo D: Modelo com bloco de convolução denso e maior contexto.

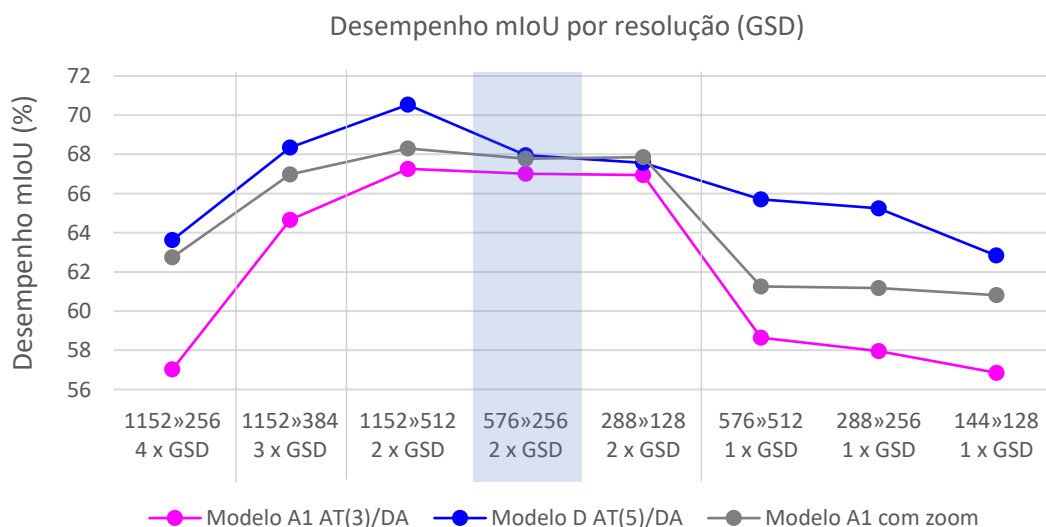
O gráfico de desempenho na Figura 6.10 mostra que, no modelo D, obter um contexto maior (por meio da subamostragem dos mapas de atributos dos blocos 4 e 5 da CNN) e, além disso, utilizar os cinco blocos originais da CNN pré-treinados e ajustados na configuração AT(5)/DA, melhora o desempenho do modelo C AT(3)/DA nas três primeiras plotagens (GSD maiores). Desta forma, os mapas de atributos de todos os blocos da CNN,

com fator de subamostragem de 4, 8, 16 e 32, contribuem para obter um maior contexto semântico em múltiplas escalas no bloco denso. Concomitantemente, os mapas de atributos de várias resoluções de todos os blocos CNN permitem recuperar a resolução da imagem original.



**Figura 6.10** – Desempenho do modelo D com bloco denso de maior contexto semântico.

Todos os treinamentos, até agora, não realizaram transformações de *zoom*, ou seja, variações de escala nos blocos de entrada, com intenção de aumentar artificialmente os blocos de treinamento em diversas escalas, o que aumentaria consideravelmente o tempo de treinamento. Embora esta transformação possa melhorar o desempenho de uma rede que não tenha uma estrutura que permita extrair atributos multiescala, pelo menos em nossos testes, o desempenho não foi melhor do que o desempenho de redes multiescala. Para exemplificar, no gráfico da [Figura 6.11](#), o modelo “A1 com *zoom*” – treinado com aumento de dados em escala (linha cinza) – embora melhore o desempenho em relação ao modelo A1 de referência (sem *zoom*), tem um desempenho inferior ao modelo D (com contexto multiescala), que foi treinado sem aumento de dados em escalas variadas. Como vimos, o desenvolvimento de redes de segmentação semântica com melhor desempenho na extração de atributos multiescala é crucial para algumas aplicações com conjunto limitado de dados rotulados em diversas escalas.



**Figura 6.11** – Desempenho do modelo de referência com aumento de dados (*zoom*).



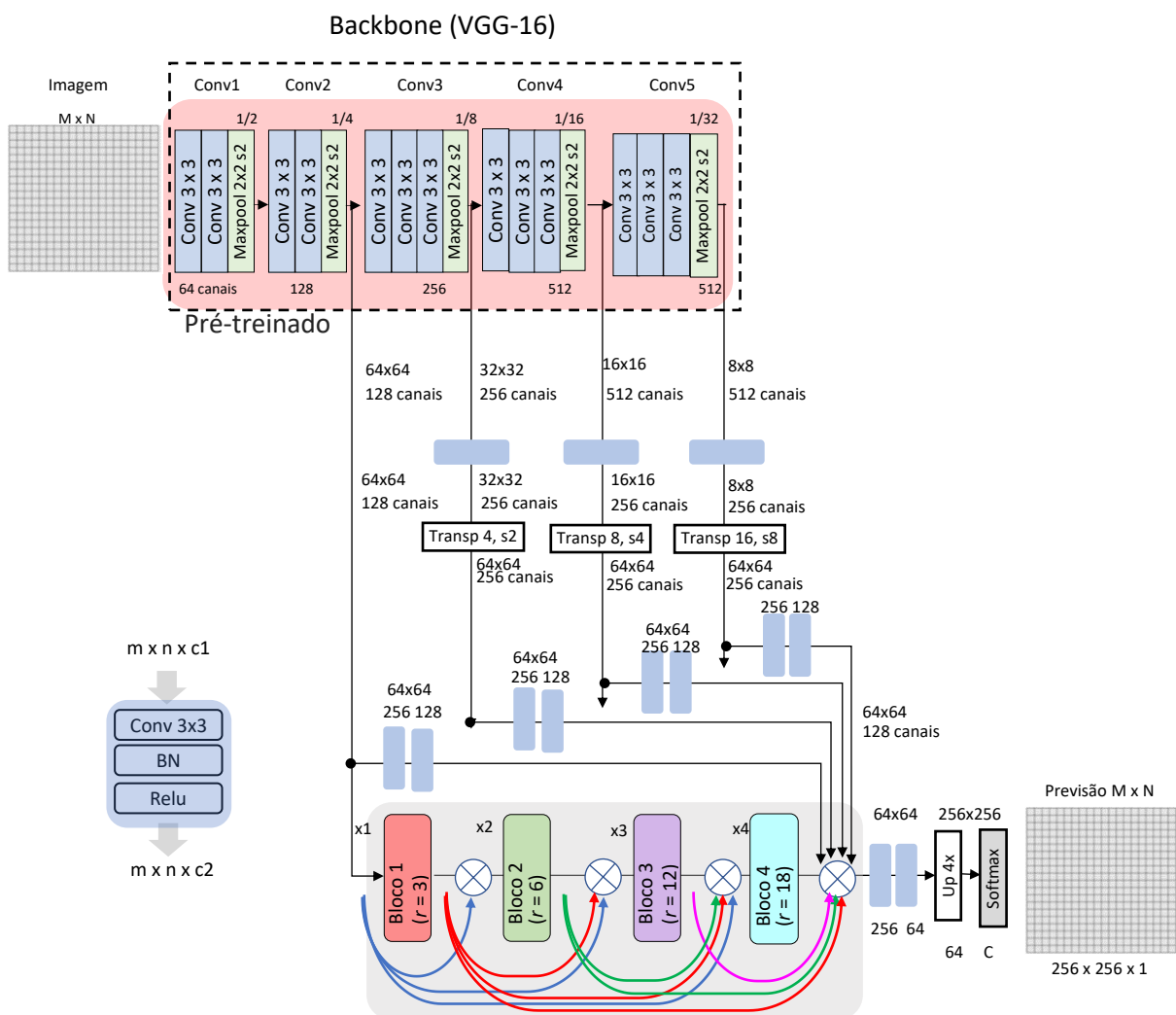
### 6.2.5 Modelos sem efeito multiescala

Testamos alguns modelos que falharam em obter atributos multiescala, mesmo usando o bloco denso. A seguir, descrevemos as características destes modelos, que fornecem informações relevantes para entender a importância de cada elemento estrutural do modelo proposto.

#### 6.2.5.1 Bloco denso paralelo sem mapas intermediários

Se removermos os mapas de atributos provenientes dos blocos intermediários do *backbone*, que estão ligados paralelamente ao bloco denso (no modelo D), o modelo E na [Figura 6.12](#) perde o efeito multiescala, obtendo desempenho inferior ao modelo A1 AT(3)/DA de referência, como mostra a [Figura 6.14](#).

Diante disso, verificamos que o bloco denso extrai atributos importantes dos blocos intermediários do *backbone*, analisando cada mapa de atributos em diversos campos de visão (FoV) para obter atributos multiescala, como será analisado na [Seção 6.4](#).



**Figura 6.12** – Modelo E: Modelo sem blocos intermediários no bloco denso.

### 6.2.5.2 Bloco denso inverso

Se invertermos o bloco denso do modelo D, ou seja, utilizando uma estrutura com fluxo de informação semelhante ao da arquitetura codificador-decodificador em U (Figura 6.1a), o modelo F na Figura 6.13 não apresenta bom desempenho multiescala, como mostra o gráfico da Figura 6.14. Além disso, com um fator de 32 no topo da CNN, blocos de entrada de dimensões pequenas (por exemplo,  $144 \times 144$ ) deterioram os filtros de convolução *atrous* com taxas de dilatação maiores em convolução  $1 \times 1$ .

Da mesma forma que o modelo F, a rede DenseASPP também usa um módulo de bloco denso no topo da CNN, porém, sem conexões de salto de mapas intermediários. Em outras palavras, o bloco denso analisa apenas os mapas de atributos de baixa resolução (com fator de 8) do bloco 5 (com deterioração de filtros *atrous*, se a dimensão da entrada não for suficiente). Entretanto, assim como o modelo F, o desempenho da rede DenseASPP é inferior ao modelo D em imagens de alta resolução, como mostra o gráfico da Figura 6.15.

Com isso em mente, em nossos experimentos, não observamos vantagem em conectar uma sequência de convoluções *atrous* com taxas  $r = [3, 6, 12, 18]$ , com a finalidade de analisar, de forma consecutiva, os mapas de atributos de baixa resolução do bloco 5, com campos receptivos densos e crescentes. Em contraposição, no modelo D verificamos que há uma melhoria de desempenho quando os blocos intermediários de maior resolução são analisados pelo bloco denso em série e em paralelo, com quatro convoluções *atrous* com taxas  $r = [3, 6, 12, 18]$ , para obter maior contexto semântico e escalas densas variadas.

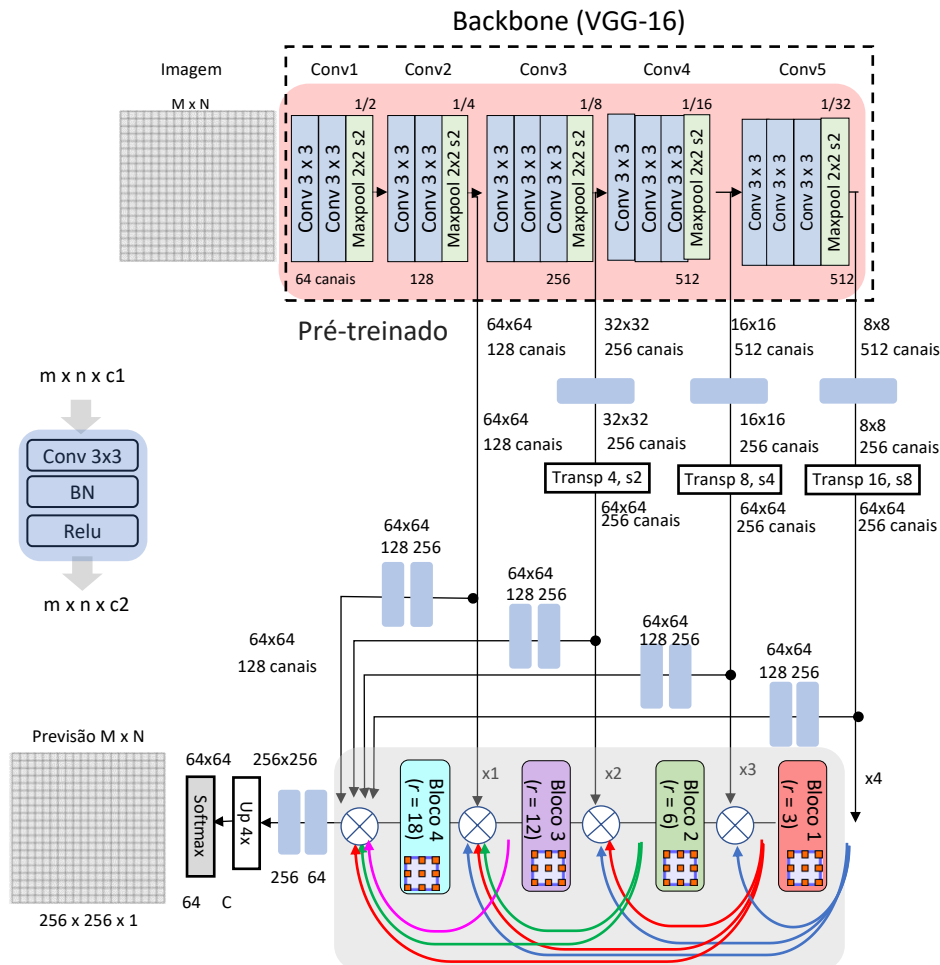


Figura 6.13 – Modelo F: Modelo com bloco denso inverso.

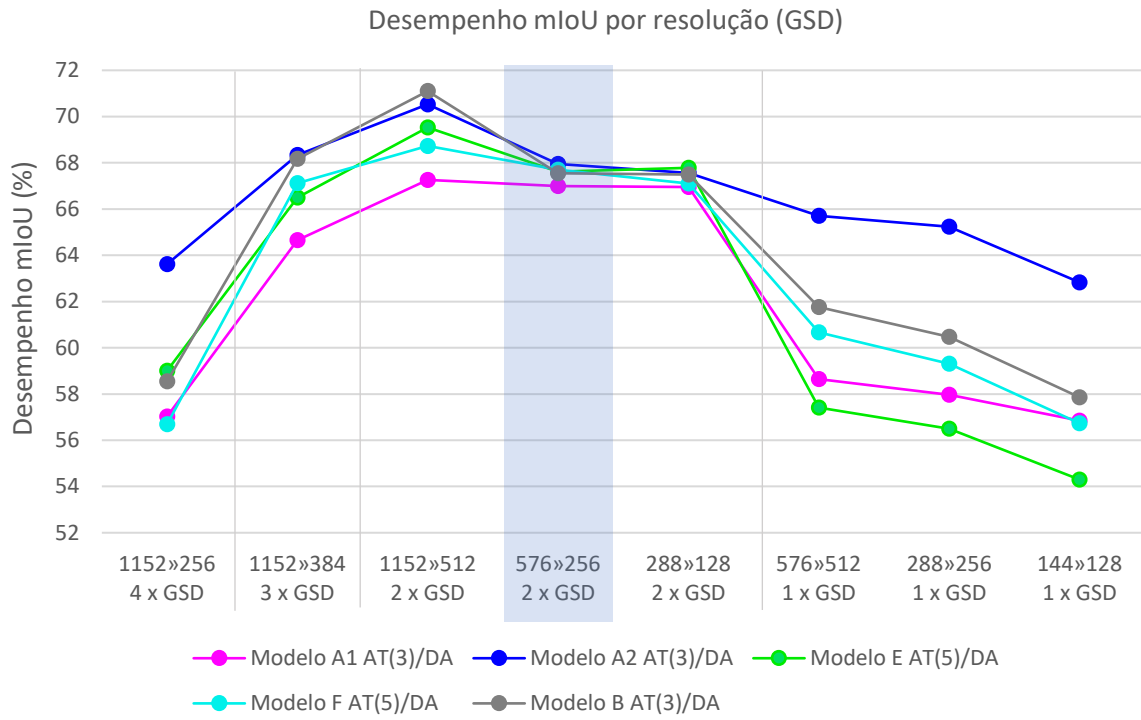


Figura 6.14 – Desempenho do modelo E sem atributos intermediários no bloco denso.

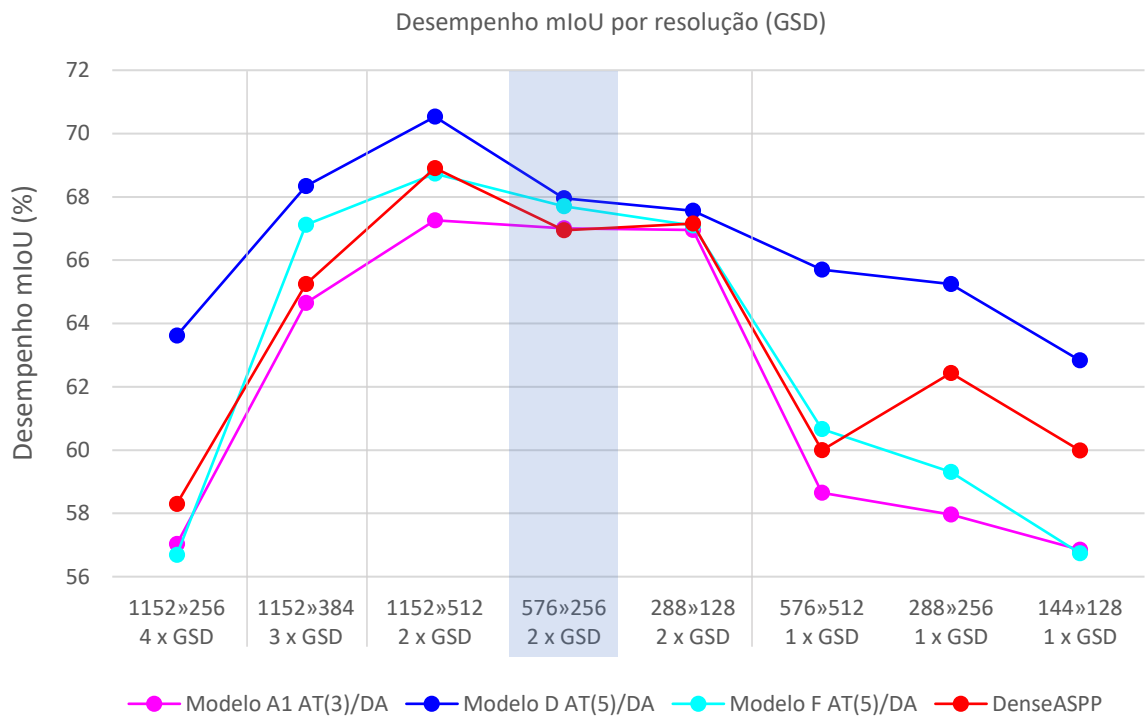


Figura 6.15 – Desempenho do modelo F e DenseASPP com bloco denso no topo da CNN.

## 6.3 Resultados quantitativos dos modelos desenvolvidos

Nesta seção, complementamos a análise quantitativa do desempenho de predição das redes desenvolvidas para segmentação semântica de mosaico de imagens de VANT. As redes foram treinadas e testadas usando o conjunto de dados de imagens aéreas RGB de uma área de cultivo de cana-de-açúcar com presença de plantas daninhas.

### 6.3.1 Resultados quantitativos com imagens aéreas RGB

Os modelos desenvolvidos foram treinados apenas na configuração AT/DA (*backbone* pré-treinado no ImageNet e com pesos reajustados no conjunto de dados de treinamento, com aumento artificial de dados) por apresentar melhor desempenho. Nos experimentos da [Tabela 6.1](#), os modelos A a C reutilizam apenas as três camadas iniciais da CNN-base pré-treinada, redefinindo as camadas restantes do *backbone* com convolução *atrous* sem subamostragem. Assim, testamos a configuração AT(3)/DA, ou seja, as três camadas iniciais são inicializadas com transferência de aprendizado e as camadas restantes são inicializadas aleatoriamente, mas todos os parâmetros são treináveis. Os modelos restantes D a F usam as cinco camadas originais da CNN, pré-treinadas e reajustadas em nosso conjunto de dados na configuração AT(5)/DA.

A [Tabela 6.1](#) mostra o número de parâmetros totais, treináveis e não treináveis, além do tempo de treinamento. Alguns experimentos foram executados em uma GPU de menor velocidade (marcadas com asterisco). Apesar do modelo A1 ter aproximadamente o mesmo número de parâmetros que o modelo B, com 19.956.884 parâmetros no total, ele apresenta maior tempo de treinamento devido às operações de convolução em mapas de atributos de maior resolução, logo após as operações de sobreamostragem com interpolação bilinear de 4x. Os modelos D a F têm mais parâmetros do que o modelo C devido à operação de convolução transposta com maior taxa de sobreamostragem.

Conseqüentemente, para desenvolver novos modelos é necessário equilibrar a quantidade de parâmetros e o número de operações da arquitetura, controlando o tamanho dos mapas de atributos, número de canais e filtros de convolução, a fim de construir modelos com menor complexidade e maior capacidade de generalização, o que evita o sobreajuste em aplicações com poucos dados de treinamento. Isso também ajuda a obter um menor tempo de treinamento, diminui os recursos computacionais necessários e permite fazer a predição em tempo quase real.

**Tabela 6.1** – Número de parâmetros e tempo de treinamento dos modelos propostos.

Modelo de Segmentação	CNN-base	Parametros total	Parametros treináveis	Parametros não treináveis	Tempo de treinamento	Configuração de rede
Modelo A1	VGG-16	19.956.884	19.954.188	2.696	*11 h 36 min	AT(3)/DA
Modelo A2	VGG-16	34.249.876	34.245.132	4.744	*15 h 05 min	AT(3)/DA
Modelo B	VGG-16	19.211.540	19.204.748	6.792	*5 h 44 min	AT(3)/DA
Modelo C	VGG-16	26.499.988	26.491.404	8.584	*6 h 36 min	AT(3)/DA
Modelo D	VGG-16	45.376.404	45.366.796	9.608	5 h 53 min	AT(5)/DA
Modelo E	VGG-16	45.275.028	45.266.956	8.072	5 h 36 min	AT(5)/DA
Modelo F	VGG-16	45.427.092	45.416.716	10.376	*6 h 36 min	AT(5)/DA

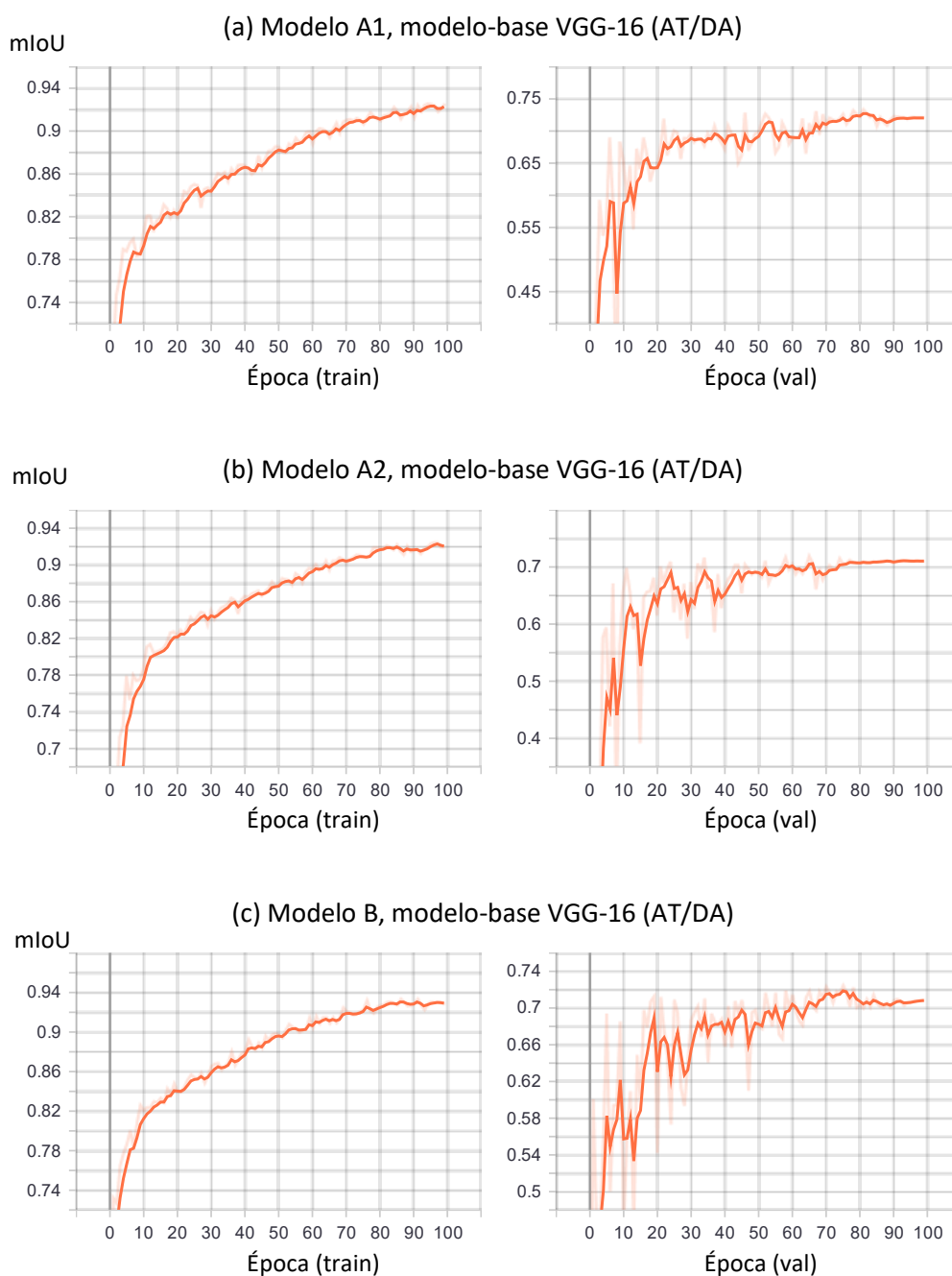
DA – Data Augmentation, AT – All Trainable

\*GPU de menor velocidade

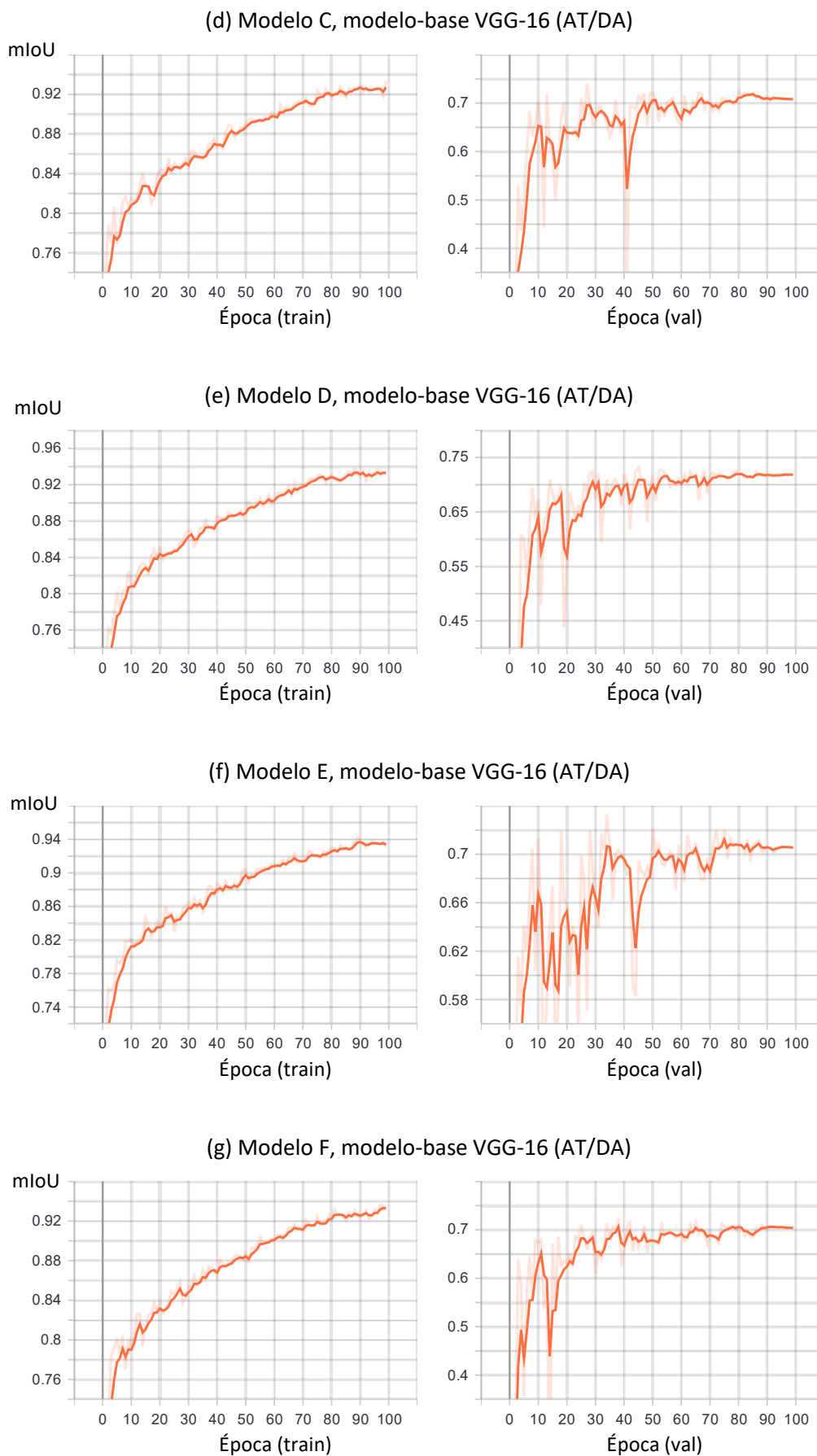
### 6.3.1.1 Medidas de desempenho de treinamento e validação

Durante o procedimento de treinamento/validação, as medidas de desempenho mIoU de validação foram monitoradas nas três imagens RGB que compõem o conjunto de validação. Na [Figura 6.16](#) podemos ver o gráfico de desempenho (mIoU  $\times$  época) de treinamento e validação de cada modelo desenvolvido.

Como todos os modelos têm muitos parâmetros treináveis, o treinamento foi executado por 100 épocas para evitar sobreajuste excessivo e problemas de generalização no conjunto de teste. Observa-se que todas as redes convergiram rapidamente no início do treinamento, mas embora o desempenho continue aumentando, com IoU médio acima de 90%, o desempenho de validação não ultrapassa 72% devido à quantidade limitada de dados para treinamento da rede.







**Figura 6.16** – Tensorboard: desempenho mIoU de treinamento e validação.

### 6.3.1.2 Medidas de desempenho de teste

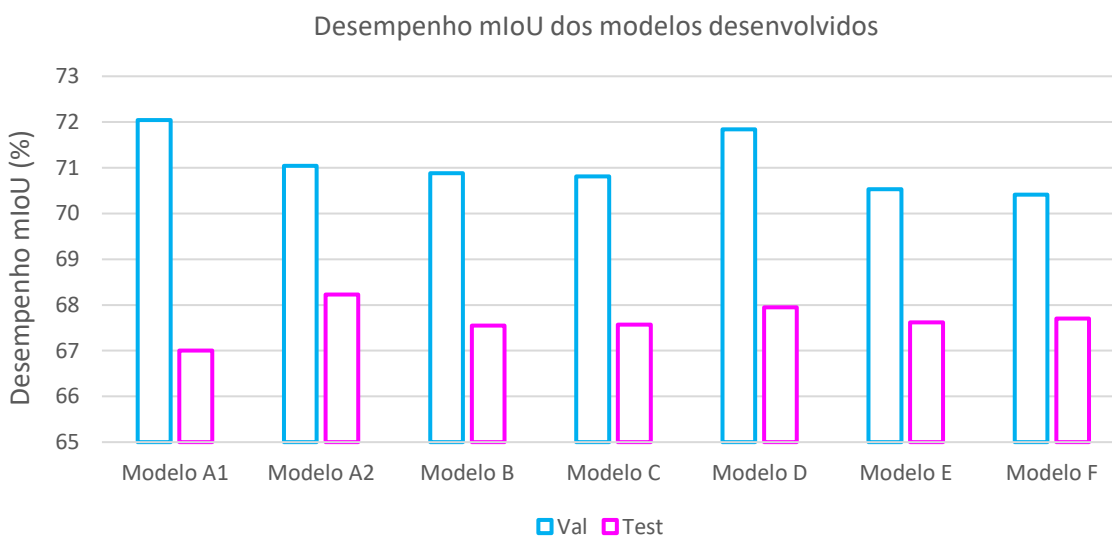
Após o treinamento, foram realizadas as avaliações no conjunto de blocos de teste de três imagens aéreas originais RGB. Os resultados quantitativos de mIoU de validação e teste (com pesos treinados na época 100) são apresentados na [Tabela 6.2](#).

**Tabela 6.2** – Desempenho IoU médio dos modelos propostos no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada.

Modelo de Segmentação	CNN-base	mIoU (%)		Configuração
		Val	Test	
Modelo A1	VGG-16	72,04	67,00	AT(3)/DA
Modelo A2	VGG-16	71,04	68,23	AT(3)/DA
Modelo B	VGG-16	70,88	67,55	AT(3)/DA
Modelo C	VGG-16	70,81	67,57	AT(3)/DA
Modelo D	VGG-16	<b>71,84</b>	<b>67,95</b>	AT(5)/DA
Modelo E	VGG-16	70,53	67,62	AT(5)/DA
Modelo F	VGG-16	70,41	67,70	AT(5)/DA

DA – Data Augmentation, AT – All Trainable

O gráfico da [Figura 6.17](#) apresenta os valores mIoU de validação e teste de cada modelo na configuração AT/DA. Todos os modelos apresentaram desempenhos semelhantes, com valores de IoU médio de teste entre 67,00% e 68,23%.



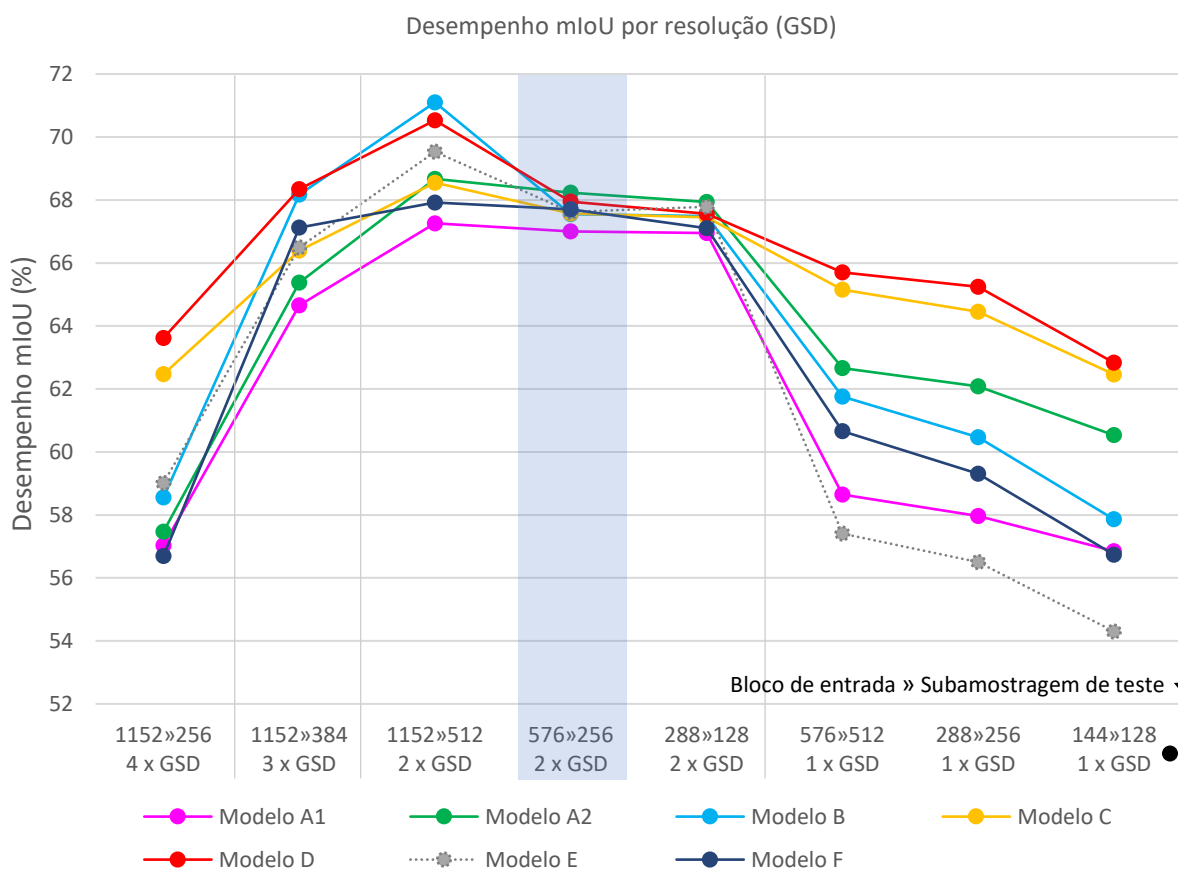
**Figura 6.17** – Desempenhos de cada modelo no conjunto de validação e teste.

Para analisar o desempenho multiescala, realizamos testes das redes em diferentes tamanhos de blocos e GSDs. A [Tabela 6.3](#) mostra os resultados de desempenho dos modelos de rede na configuração AT/DA, treinadas com blocos de tamanho  $576 \times 576$  subamostrados para  $256 \times 256$  pixels (representado por  $576 \gg 256$ ) e testados em diversos tamanhos de bloco e resoluções espaciais (GSD). A diferença de desempenho mIoU de teste – entre os modelos testados com o mesmo tamanho de  $576 \gg 256$  pixels e mesmo GSD utilizado no treinamento – foi muito pequena, como ilustrado na coluna azul da [Tabela 6.3](#) e da [Figura 6.18](#). Porém, a diferença aumenta com GSDs diferentes do treinado pela rede.

**Tabela 6.3** – Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.

Modelo de Segmentação	mIoU (%) de 3 imagens RGB de teste							
	1152 » 256 36 blocos	1152 » 384 36 blocos	1152 » 512 36 blocos	576 » 256 144 blocos	288 » 128 576 blocos	576 » 512 144 blocos	288 » 256 576 blocos	144 » 128 2304 blocos
	4 x GSD	3 x GSD	2 x GSD	2 x GSD	2 x GSD	1 x GSD	1 x GSD	1 x GSD
Modelo A1	57,03	64,65	67,26	67,00	66,95	58,65	57,96	56,85
Modelo A2	57,47	65,38	68,67	68,23	67,94	62,66	62,08	60,53
Modelo B	58,55	68,17	71,10	67,55	67,49	61,76	60,47	57,86
Modelo C	62,47	66,39	68,55	67,57	67,45	65,15	64,45	62,46
Modelo D	63,62	68,34	70,53	67,95	67,56	65,70	65,24	62,83
Modelo E	59,01	66,49	69,53	67,62	67,78	57,41	56,50	54,30
Modelo F	56,69	67,12	68,73	67,70	67,1	60,66	59,31	56,74

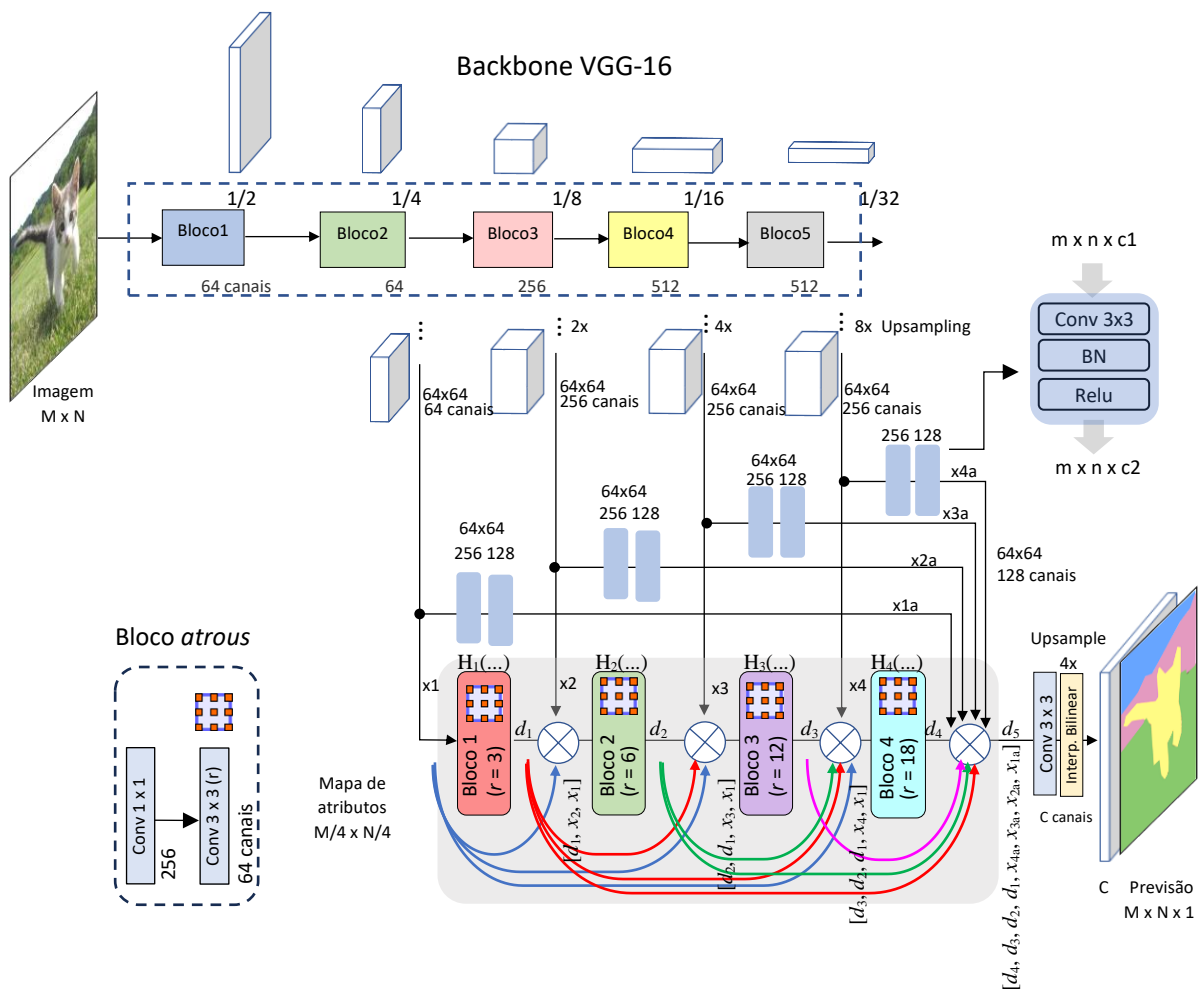
Na [Figura 6.18](#), podemos ver o gráfico correspondente à [Tabela 6.3](#), comparando o desempenho multiescala de todos os modelos desenvolvidos. O modelo D apresenta melhores resultados considerando todas as dimensões e resoluções de entrada. Por este motivo, selecionamos esta arquitetura de rede como modelo proposto para extração de atributos multiescala, com a finalidade de segmentação de plantas daninhas de diferentes tamanhos em uma cultura de cana-de-açúcar, usando mosaicos de imagens aéreas de VANT com variações de resolução espacial para aplicações de agricultura de precisão.

**Figura 6.18** – Comparação de desempenho das redes com diferentes resoluções de entrada.

### 6.4 Modelo proposto

O modelo proposto usa um bloco denso de convoluções *atrous* para extração de atributos multiescala dos mapas intermediários do *backbone*, como mostra a [Figura 6.19](#). Desta forma, atributos multiescala são extraídos no bloco denso, sem precisar treinar a rede com entrada em escalas variadas, seja alimentando-a com blocos de imagens em várias escalas, ou treinando-a com aumento artificial de dados com operação de *zoom*, o que aumenta o tempo de treinamento. Além disso, os atributos de alta resolução são preservados e concatenados na saída do bloco denso, com intuito de recuperar a localização e as estruturas finas de objetos pequenos.

No módulo ASPP, quando as imagens são de alta resolução, a convolução *atrous* deve ter uma taxa de dilatação grande para capturar campos receptivos maiores. No entanto, conforme esta taxa aumenta, a convolução *atrous* torna-se cada vez mais ineficaz, com *kernels* muito esparsos. Portanto, utilizamos um bloco denso paralelo ao *backbone*, capaz de simultaneamente codificar informações multiescala e obter um campo receptivo suficientemente grande nas imagens de alta resolução, sem precisar aumentar a taxa de dilatação que deteriora os filtros de convolução *atrous*.



**Figura 6.19** – Modelo proposto com bloco denso paralelo ao backbone.

As respostas de ativação dos mapas de atributos dos blocos superiores da CNN codificam o contexto semântico de longo alcance (contexto global), enquanto os blocos inferiores preservam os detalhes de pequenos objetos (contexto local). Para englobar todo o contexto, os mapas de atributos dos ramos paralelos da CNN são interpolados bilinearmente para uma mesma resolução de 1/4 da imagem original e, posteriormente, concatenados em mapas de atributos multiescala dentro da pirâmide do bloco denso.

A Figura 6.20 mostra um bloco denso composto de  $L = 4$  blocos convolucionais *atrous*, que implementam uma função composta não linear  $H_l(\cdot)$ , onde  $l$  é o índice do bloco convolucional de 1 a  $L$ . Assim como na rede DenseASPP, a função  $H_l(\cdot)$  é definida por duas operações consecutivas: uma convolução  $1 \times 1$  para reduzir o número de mapas de atributos de entrada para 256 canais e uma convolução *atrous*  $3 \times 3$  com taxa de dilatação  $r$  e redução para 64 canais. Estes redutores de dimensão de canais evitam um aumento da complexidade computacional devido à concatenação de mapas de atributos no bloco denso.

Considerando um mapa de atributos  $x_1$  na entrada do bloco denso, a saída do bloco convolucional  $l$  é denotada por  $d_l$ . Existem conexões diretas de qualquer bloco para todos os blocos posteriores. Consequentemente, o bloco  $l$  recebe como entrada os mapas de atributos de todos os blocos anteriores e um mapa de atributos do bloco intermediário da CNN,  $x_l$  onde  $[x_1, \dots, d_1, \dots, d_{l-1}]$  refere-se à concatenação dos mapas de atributos dos blocos convolucionais  $0, \dots, l-1$ , que produz a saída  $d_l = H_l([x_1, \dots, d_1, \dots, d_{l-1}])$ .

Um bloco convolucional *atrous*  $l$  tem como entrada os mapas de atributos de todos os blocos convolucionais anteriores, e cada bloco acrescenta apenas um pequeno conjunto de  $k$  mapas de atributos (taxa de crescimento), do mapa  $d_l$  na saída do bloco  $l$ , mantendo os mapas dos blocos anteriores inalterados.

A Figura 6.20 mostra um módulo denso com quatro convoluções *atrous* em série, com *kernel*  $3 \times 3$  e taxa de dilatação  $r \leq 18$ , ou seja,  $r = [3, 6, 12 \text{ e } 18]$ . Devido à série de convoluções *atrous* consecutivas, onde a taxa de dilatação aumenta a cada bloco, os campos receptivos são cada vez maiores, sem o problema de degradação do *kernel*. Assim, os mapas de atributos de alta resolução do bloco 2 da CNN são analisados com maior campo de visão (FoV), enquanto os mapas mais profundos da CNN de menor resolução são inseridos no bloco denso e analisados com menor campo de visão.

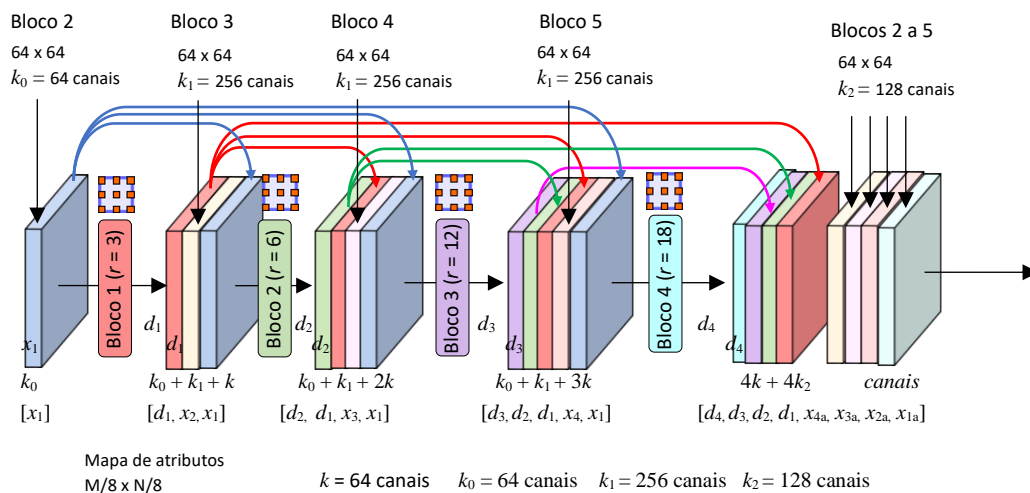


Figura 6.20 – Mapas de atributos do bloco denso no modelo proposto.



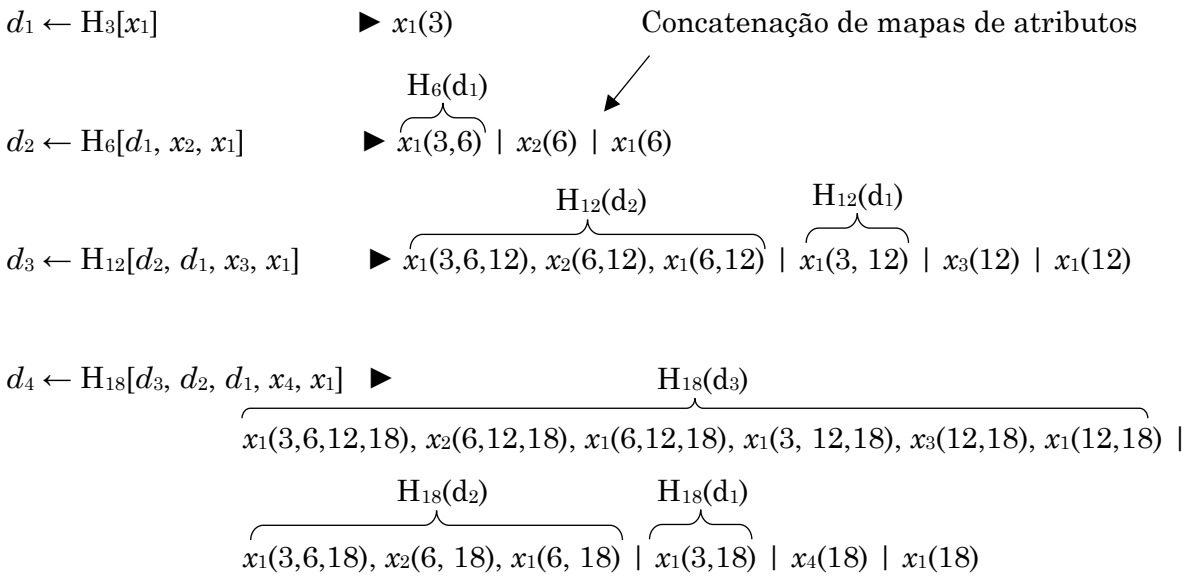
Diferente da rede DenseASPP, o mapa de entrada do módulo denso,  $x_1$ , é a saída do bloco 2 da CNN-base, com fator de redução de resolução espacial de 4. Outros mapas de atributos da CNN,  $x_l$ , servem como entrada de blocos densos intermediários. Cada bloco *atrous* é representado por:

$$d_l = H_{r_l}([d_{l-1}, d_{l-2}, \dots, x_1])$$

onde  $d_l$  é a saída do bloco  $l$ ,  $r_l$  representa a taxa de dilatação do bloco  $l$ , e  $[\cdot \cdot \cdot]$  denota a operação de concatenação de mapa de atributos.

Assim, a entrada  $x_1$  (mapa de alta resolução do bloco 2 da CNN) passa pelas camadas de convoluções *atrous*  $H_3$ ,  $H_6$ ,  $H_{12}$  e  $H_{18}$  com taxas de dilatação  $r = [3, 6, 12 \text{ e } 18]$  em série e em paralelo. A entrada  $x_2$  (mapa sobreamostrado proveniente do bloco 3) passa pelas convoluções  $H_6$ ,  $H_{12}$  e  $H_{18}$ , e assim por diante. O mapa de menor resolução da CNN,  $x_4$ , com contexto de longo alcance, passa apenas por uma camada de convolução  $H_{18}$ .

Desta forma, em cada saída  $d_l$  de uma camada convolucional *atrous*  $l$  do bloco denso, temos as entradas  $x(r)$  analisadas pelas seguintes taxas de convolução  $r$  em série (entre parênteses) e em paralelo por concatenação (barra vertical).



Diferentemente do bloco denso em DenseNet, a saída do bloco denso na rede proposta é a concatenação das saídas das quatro camadas ( $[d_4, d_3, d_2, d_1]$ ) sem a entrada  $d_0$  e, portanto, gera  $4 \times k$  mapas de atributos. Além disso, a saída do bloco denso  $[d_4, d_3, d_2, d_1]$  é concatenada com os quatro mapas  $[x_{4a}, x_{3a}, x_{2a}, x_{1a}]$  provenientes dos blocos 2 a 5 da CNN, após sobreamostragem e duas convoluções  $3 \times 3$  com redução de canais para  $k_2=128$ , adicionando  $4 \times k_2$  mapas de atributos. Assim, o mapa de atributos de saída do bloco denso é dado pela concatenação de mapas de atributos analisados em várias escalas:

$$d_5 \leftarrow [d_4, d_3, d_2, d_1, x_{4a}, x_{3a}, x_{2a}, x_{1a}]$$

Desta forma, cada mapa de atributos  $x_l$  de um bloco da CNN é analisado em várias escalas com campos de visão máximos diferentes, como detalhado abaixo. O campo de visão  $FoV_{k,r}$  de uma camada convolucional *atrous* com taxa de dilatação  $r$  e tamanho de *kernel*  $k$  é calculado pelas Equações 2.7 e 2.8.

- Mapa de atributos  $x_1$  (FoV max = 79), correspondente à pirâmide da Figura 6.21.

$x_1(3), x_1(6), x_1(3,6), x_1(12), x_1(3, 12), x_1(18), x_1(6,12), x_1(3,18), x_1(3,6,12), x_1(6, 18),$   
 $x_1(3,6,18), x_1(12,18), x_1(3, 12,18), x_1(6,12,18), x_1(3,6,12,18);$

$$FoV_{max} = FoV_{3,3} + FoV_{3,6} + FoV_{3,12} + FoV_{3,18} - 3 = 7 + 13 + 25 + 37 - 3 = 79$$

- Mapa de atributos  $x_2$  (FoV max = 73)

$x_2(6), x_2(6,12), x_2(6, 18), x_2(6,12,18)$

$$FoV_{max} = FoV_{3,6} + FoV_{3,12} + FoV_{3,18} - 2 = 13 + 25 + 37 - 2 = 73$$

- Mapa de atributos  $x_3$  (FoV max = 61)

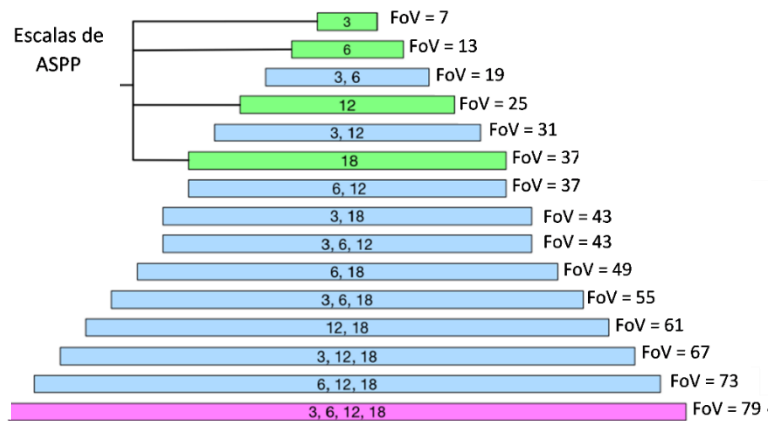
$x_3(12), x_3(12,18)$

$$FoV_{max} = FoV_{3,12} + FoV_{3,18} - 1 = 25 + 37 - 1 = 61$$

- Mapa de atributos  $x_4$  (FoV max = 37)

$x_4(18)$

$$FoV_{max} = FoV_{3,18} = 37$$



**Figura 6.21** – Pirâmide de escalas do mapa de alta resolução do modelo proposto.

Diferentemente do bloco denso com convoluções regulares  $3 \times 3$  da rede de classificação de imagens DenseNet na Figura 2.47, a convolução *atrous* permite ajustar o campo de visão da rede. A combinação dos filtros *atrous* exploram a imagem em campos receptivos variados com maior campo de visão, capturando detalhes em diferentes escalas e agregando as informações de contexto para abstrair atributos de diferentes escalas nos próximos estágios. Assim, a saída final da rede é um mapa de atributos gerado por convoluções *atrous* de múltiplas taxas e multiescala (diferentes tamanhos de campos receptivos).

Na Figura 6.9 e Figura 6.19, a arquitetura proposta tem um *backbone* idêntico às camadas convolucionais da rede VGG-16 (sem as camadas FC) e, portanto, usa as cinco camadas convolucionais pré-treinadas no conjunto de dados de classificação ImageNet. Outras redes classificadoras podem ser usadas como base, mas seus desempenhos não foram avaliados neste trabalho.

A informação mais abstrata e global das camadas mais profundas da CNN-base abstrai melhor o contexto semântico, enquanto a informação local de maior resolução das camadas mais rasas tem maior precisão nas estruturas finas de objetos. Assim, a rede proposta usa diretamente os mapas de atributos do *backbone* no bloco denso, sem subamostragem, para evitar perda de informação. Devido às operações consecutivas de subamostragem na CNN-base, os mapas de atributos dos blocos intermediários e de saída têm diferentes resoluções espaciais. Portanto, os mapas de atributos dos três blocos superiores são sobreamostrados com interpolação bilinear ou convolução transposta de 2x, 4x e 8x, antes de serem concatenados no bloco denso. Por fim, esses mapas sobreamostrados são concatenados com a saída do bloco denso, combinando informações de múltiplas resoluções da hierarquia de atributos das camadas intermediárias, que codificam conjuntamente a localização e a semântica, para refinar a precisão espacial do mapa de segmentação e melhorar o desempenho da predição densa.

Como o fator de redução da resolução espacial na saída do bloco denso é 4, o mapa de atributos é sobreamostrado com uma interpolação bilinear de 4x para recuperar a resolução original. Na camada final, uma convolução  $1 \times 1$  é usada para mapear cada mapa de atributos sobreamostrado de 64 canais para um número desejado de classes. Assim, na última camada convolucional, o número de mapas de atributos corresponde à quantidade de classes, onde cada mapa representa as probabilidades espaciais em relação a uma determinada classe.

Na última etapa, os mapas de atributos de alta resolução são alimentados em uma camada *softmax* para prever um mapa de probabilidade de classe *pixel* a *pixel*. Para cada *pixel* correspondente nos mapas de atributos, a classe com maior probabilidade ao longo de todos os canais é usada como rótulo previsto, e os rótulos previstos de todos os *pixels* na imagem formam o mapa de previsão de saída.

Nossa arquitetura de salto – DenseLensNet – com um bloco denso de convoluções *atrous* paralelo ao *backbone*, permite combinar e analisar em multiescala as informações semânticas abstratas mais complexas (global) e refinadas (local) com diferentes campos de visão simultaneamente.



# Capítulo 7

## Análise e comparação de resultados

Neste capítulo, fazemos uma comparação do desempenho quantitativo e qualitativo do nosso modelo proposto em relação a alguns modelos multiescala selecionados, que foram avaliados no [Capítulo 5](#), usando imagens aéreas de teste e um mosaico de imagens de VANT de uma cultura de cana-de-açúcar.

### 7.1 Resultados quantitativos com imagens aéreas RGB

Primeiro, os modelos foram treinados e testados nas imagens aéreas RGB com tamanho de bloco de  $576 \times 256$  pixels e GSD médio de 1,16 cm/pixel, conforme relatado no [Capítulo 5](#) e [Capítulo 6](#).

A [Tabela 7.1](#) mostra o número de parâmetros totais, treináveis e não treináveis, além do tempo de treinamento de cada rede. Nosso modelo com *backbone* VGG-16 tem o maior número de parâmetros totais treináveis devido ao uso da convolução transposta, mas o menor tempo de treinamento, com exceção da rede BiSeNet.

A rede BiSeNet e o modelo proposto usam um *backbone* com as cinco camadas semelhantes à CNN-base original. Desta forma, na configuração AT(5), todas as camadas do *backbone* são inicializadas por transferência de aprendizado e todos os parâmetros são reajustados.

As redes DeepLab-v3, DeepLab-v3+ e DenseASPP reutilizam apenas 3 ou 4 camadas iniciais da CNN original, redefinindo as camadas restantes do *backbone* com convolução *atrous* sem subamostragem. Assim, testamos as configurações AT(3) ou AT(4), dependendo da arquitetura da rede, onde apenas as  $n$  primeiras camadas do *backbone* são transferidas, mas todos os parâmetros são reajustados.

**Tabela 7.1** – Número de parâmetros e tempo de treinamento dos modelos.

Modelo de Segmentação	CNN-base	Parametros total	Parametros treináveis	Parametros não treináveis	Tempo de treinamento	Configuração de rede
DeepLab-v3	ResNet-50	39.123.332	39.069.700	53.632	25 h 40 min	AT(4)/DA
DeepLab-v3+	ResNet-50	40.428.660	40.373.908	54.752	26 h 50 min	AT(4)/DA
DenseASPP	ResNet-50	27.169,988	27.092.548	77.440	8 h 43 min	AT(3)/DA
BiSeNet	Xception	26.081.488	26.020.504	60.984	2 h 59 min	AT(5)/DA
Modelo D	VGG-16	45.376.404	45.366.796	9.608	5h 53 min	AT(5)/DA



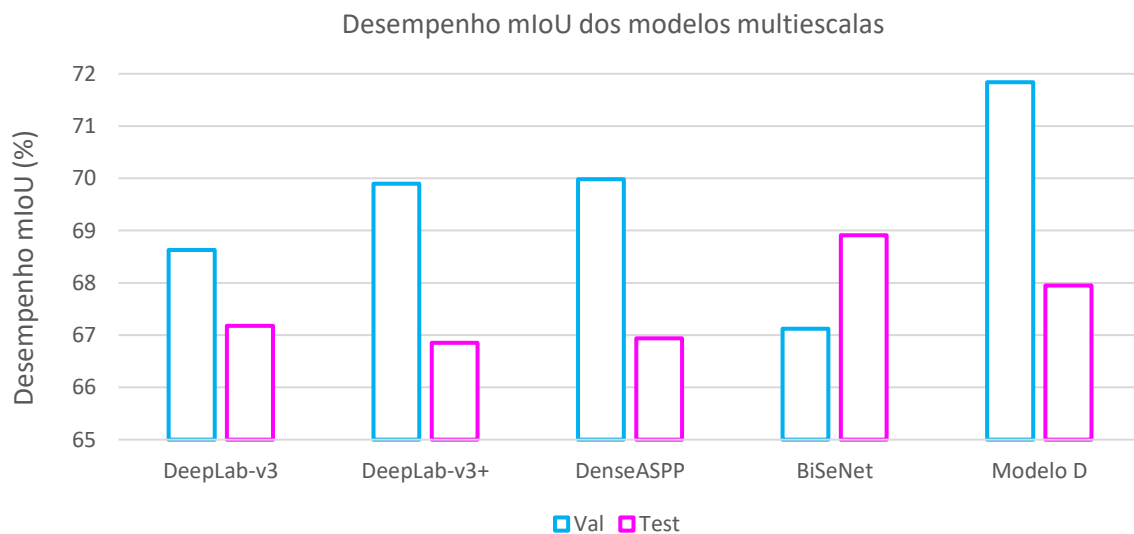
### 7.1.1 Medidas de desempenho de validação e teste

Considerando os experimentos já realizados nos dois capítulos anteriores, ou seja, com treinamento e teste com o mesmo tamanho de bloco  $576 \times 256$  pixels e mesmo GSD, a Tabela 7.2 e o gráfico da Figura 7.1 re representam os valores mIoU de validação e teste para comparação. Todos os modelos apresentaram desempenhos semelhantes, com valores de IoU médio de teste entre 65,32% e 68,91%. Em comparação com a rede BiSeNet, que obteve o melhor desempenho de IoU médio de teste, o resultado do modelo proposto foi ligeiramente inferior, mas superou os resultados das redes DeepLab e DenseASPP.

**Tabela 7.2** – Desempenho IoU médio dos modelos no conjunto de validação e teste composto por blocos de três imagens aéreas RGB cada.

Modelo de Segmentação	CNN-base	mIoU (%)		Configuração
		Val	Test	
DeepLab-v3	ResNet-50	68,63	67,18	AT(4)/DA
DeepLab-v3+	ResNet-50	69,90	66,85	AT(4)/DA
DenseASPP	ResNet-50	69,98	66,94	AT(3)/DA
BiSeNet	Xception	67,12	<b>68,91</b>	AT(5)/DA
Modelo D	VGG-16	71,84	<b>67,95</b>	AT(5)/DA

DA – Data Augmentation, AT – All Trainable



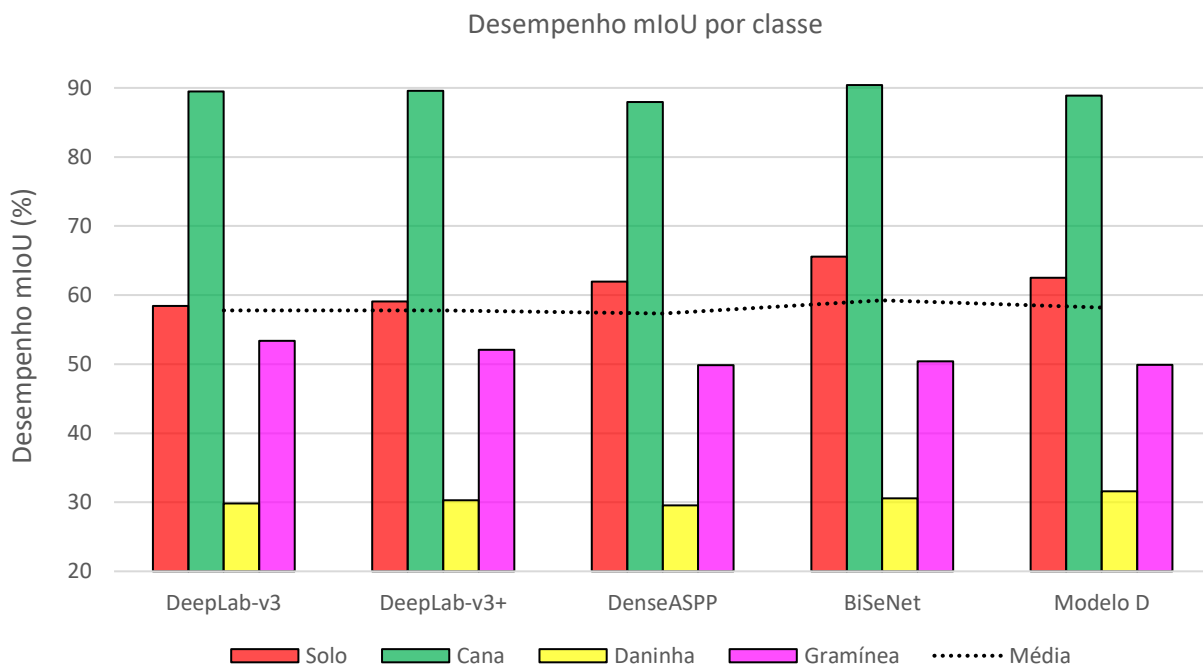
**Figura 7.1** – Desempenho do modelo proposto, comparado aos modelos multiescalas, no conjunto de validação e teste das imagens aéreas.

Os resultados quantitativos de IoU médio por classe no conjunto de teste das imagens aéreas são apresentados na Tabela 7.3 e mostrados no gráfico da Figura 7.2. A rede proposta teve um desempenho médio em todas as classes comparável às redes selecionadas e testadas neste trabalho. A média da classe daninha do modelo proposto é ligeiramente maior do que outros modelos e a média global é ligeiramente superior quando comparada com os métodos DeepLab e DenseASPP.

**Tabela 7.3** – Desempenho de *mIoU* por classe, no conjunto de teste composto por blocos de três imagens aéreas RGB.

Modelo de Segmentação	CNN-base	Solo %	Cana %	Daninha %	Gramínea %	IoU médio	Configuração
DeepLab-v3	ResNet-50	58,43	89,47	29,81	<b>53,39</b>	57,77	AT(4)/DA
DeepLab-v3+	ResNet-50	59,10	89,60	30,28	52,10	57,77	AT(4)/DA
DenseASPP	ResNet-50	61,96	87,94	29,54	49,85	57,33	AT(3)/DA
BiSeNet	Xception	<b>65,57</b>	<b>90,39</b>	30,57	50,42	<b>59,24</b>	AT(5)/DA
Modelo D	VGG-16	62,49	88,86	<b>31,58</b>	49,89	58,20	AT(5)/DA

DA – Data Augmentation, AT – All Trainable

**Figura 7.2** – Desempenhos *mIoU* por classe, no conjunto de teste composto por blocos de três imagens aéreas RGB.

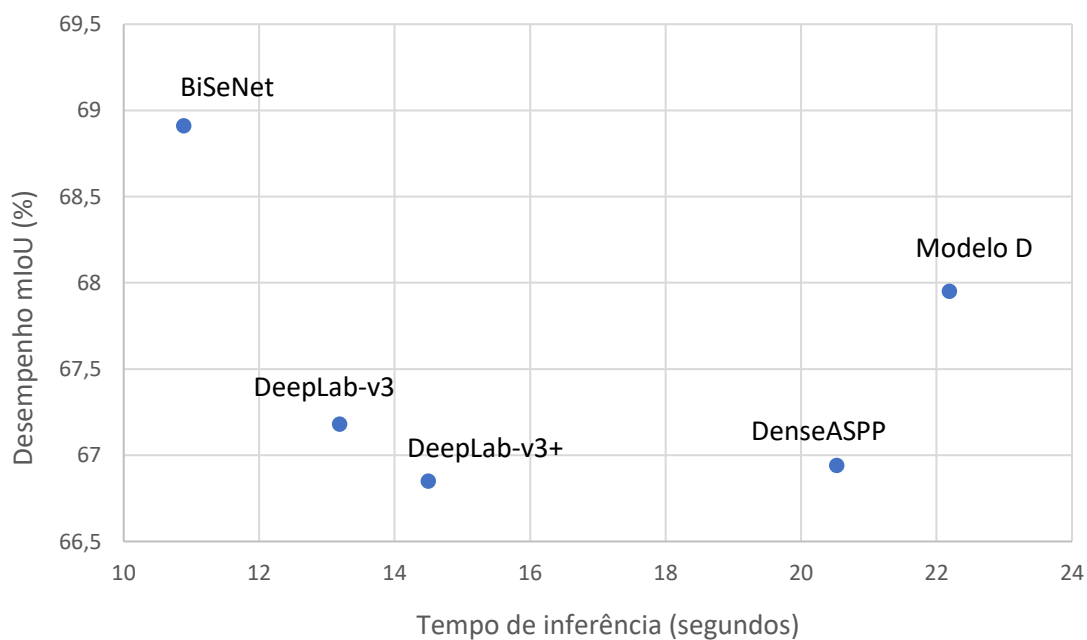
A [Tabela 7.4](#) mostra os resultados de desempenho dos modelos de rede na configuração AT/DA, treinadas com blocos de tamanho  $576 \times 256$  pixels, e testados em diferentes tamanhos de blocos e GSDs a partir de três imagens aéreas de teste. A [Tabela 7.5](#) mostra o tempo correspondente de predição dos blocos. O gráfico da [Figura 7.3](#) plota o desempenho versus tempo de inferência dos blocos de teste de  $576 \times 256$ , que corresponde à coluna azul na [Tabela 7.4](#) e [Tabela 7.5](#), respectivamente. Observamos que a rede BiSeNet apresenta o menor tempo de execução de teste, portanto, além de melhor desempenho, tem maior velocidade para predição em tempo real. Nosso modelo proposto e a rede DenseASPP são mais lentos devido ao número de operações no bloco denso.

**Tabela 7.4** – Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.

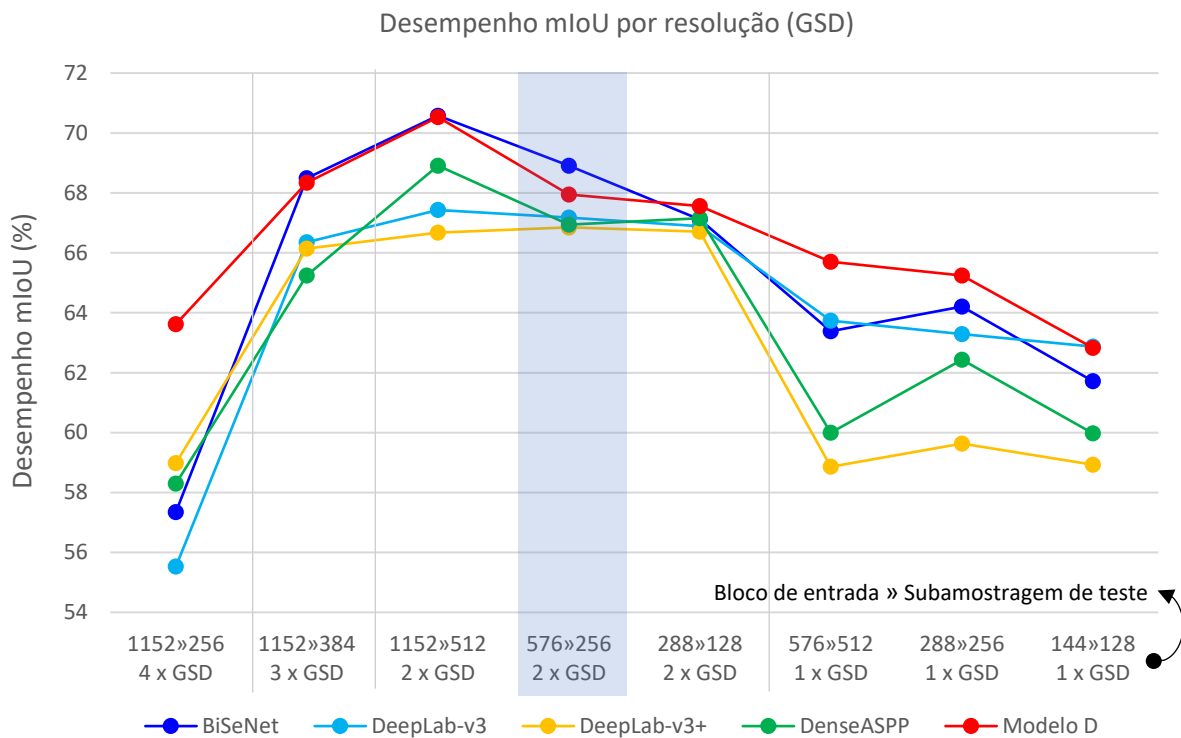
Modelo de Segmentação	mIoU (%) de 3 imagens RGB de teste							
	1152 » 256 36 blocos	1152 » 384 36 blocos	1152 » 512 36 blocos	576 » 256 144 blocos	288 » 128 576 blocos	576 » 512 144 blocos	288 » 256 576 blocos	144 » 128 2304 blocos
	4 x GSD	3 x GSD	2 x GSD	2 x GSD	2 x GSD	1 x GSD	1 x GSD	1 x GSD
DeepLab-v3	55,53	66,35	67,43	67,18	66,89	63,74	63,29	62,87
DeepLab-v3+	58,98	66,14	66,68	66,85	66,71	58,86	59,63	58,93
DenseASPP	58,30	65,24	68,91	66,94	67,16	60,00	62,43	59,98
BiSeNet	57,35	68,49	70,58	68,91	67,11	63,38	64,21	61,72
Modelo D	63,62	68,34	70,53	<b>67,95</b>	67,56	65,70	65,24	62,83

**Tabela 7.5** – Tempo de execução dos testes das redes com diferentes resoluções de entrada.

Modelo de Segmentação	Tempo de teste (em segundos) de 3 imagens RGB (segundos)							
	1152 » 256 36 blocos	1152 » 384 36 blocos	1152 » 512 36 blocos	576 » 256 144 blocos	288 » 128 576 blocos	576 » 512 144 blocos	288 » 256 576 blocos	144 » 128 2304 blocos
	4 x GSD	3 x GSD	2 x GSD	2 x GSD	2 x GSD	1 x GSD	1 x GSD	1 x GSD
DeepLab-v3	4,84	14,11	8,57	13,19	20,13	32,22	40,21	74,81
DeepLab-v3+	5,04	7,35	10,33	14,50	22,32	37,69	44,10	80,71
DenseASPP	6,35	11,58	16,89	20,53	18,11	69,30	68,09	67,02
BiSeNet	4,05	11,66	8,48	10,89	8,420	32,45	35,78	30,98
Modelo D	6,16	10,87	17,08	22,19	19,07	70,93	83,08	71,73

**Figura 7.3** – Tempo de inferência (segundos) e desempenho mIoU (%) no conjunto de três imagens aéreas de teste (144 blocos de 576 » 256).

Na [Figura 7.4](#), podemos ver o gráfico correspondente à [Tabela 7.4](#), comparando o desempenho multiescala do modelo proposto em relação a outras redes. Todos os modelos apresentam pouca diferença de desempenho quando o tamanho do bloco e GSD são os mesmos do treinamento (coluna azul). Nas outras colunas, onde o GSD é diferente do treinamento, o modelo proposto apresenta melhores resultados, em geral superando todas as redes multiescalas testadas neste trabalho.

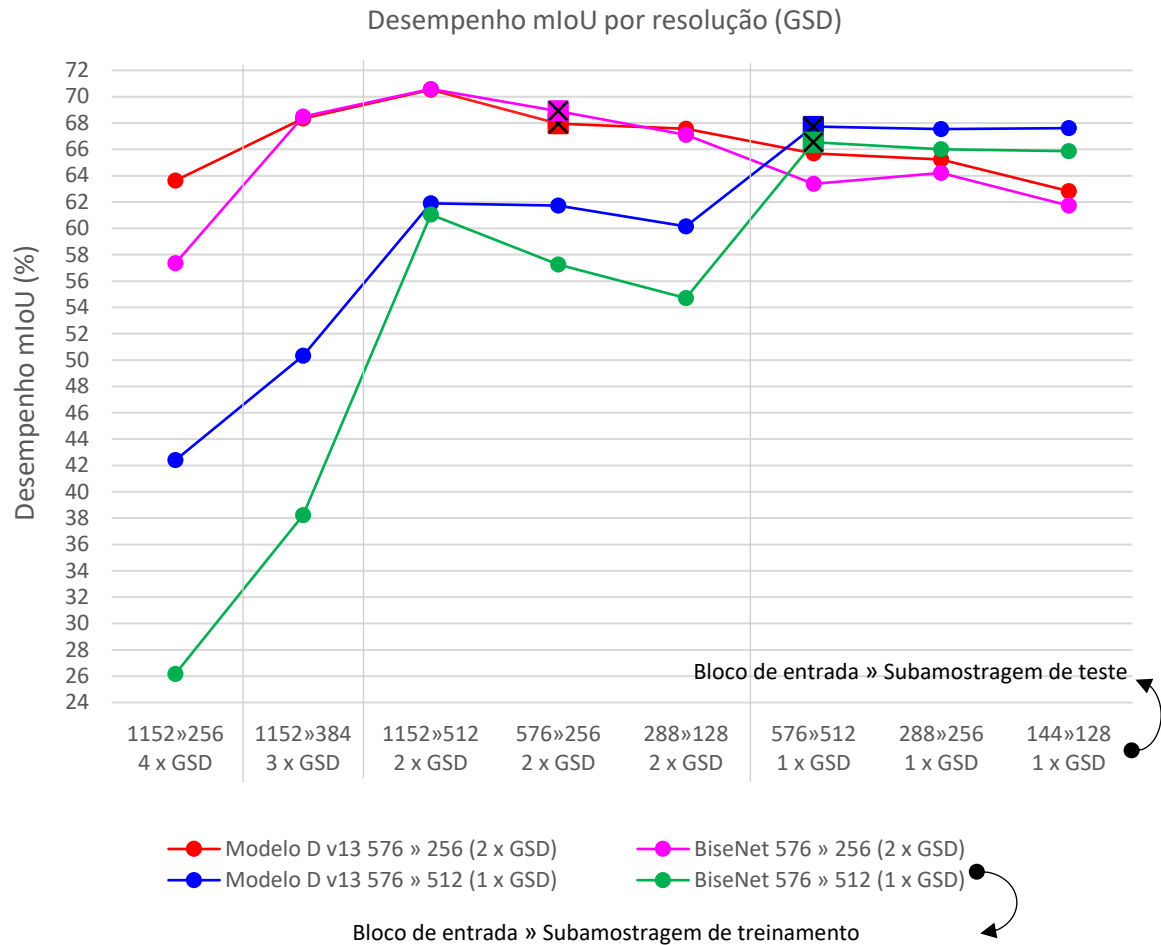


**Figura 7.4** – Comparação do modelo proposto com as redes de melhor desempenho.

A [Tabela 7.6](#) e o gráfico da [Figura 7.5](#) comparam o modelo proposto e o modelo BiSeNet treinados com tamanho de bloco 576 » 256 (2,25 x GSD) e 576 » 512 (1,125 x GSD). O modelo proposto supera o modelo BiSeNet quando treinado com imagens de alta resolução (1 x GSD) com GSD médio de 1,16 cm/pixel, principalmente quando o GSD é diferente do qual a rede foi treinada. No entanto, ainda há uma perda significativa de desempenho quanto mais distante a resolução de teste de entrada estiver do GSD treinado pela rede (mostrado com destaque na tabela e marcador quadrado no gráfico).

**Tabela 7.6** – Desempenho de teste das redes (AT/DA) com diferentes resoluções de entrada.

Modelo de Segmentação (tamanho de blocos e GSD de treinamento)	mIoU (%) de 3 imagens RGB de teste							
	1152 » 256 36 blocos	1152 » 384 36 blocos	1152 » 512 36 blocos	576 » 256 144 blocos	288 » 128 576 blocos	576 » 512 144 blocos	288 » 256 576 blocos	144 » 128 2304 blocos
	4 x GSD	3 x GSD	2 x GSD	2 x GSD	2 x GSD	1 x GSD	1 x GSD	1 x GSD
BiSeNet 576»256 (2 x GSD)	57,35	68,49	70,58	68,91	67,11	63,38	64,21	61,72
BiSeNet 576»512 (1 x GSD)	26,18	38,22	61,04	57,26	54,70	66,54	66,01	65,85
Modelo D 576»256 (2 x GSD)	63,62	68,34	70,53	67,95	67,56	65,70	65,24	62,83
Modelo D 576»512 (1 x GSD)	42,40	50,33	61,90	61,72	60,14	67,73	67,54	67,62



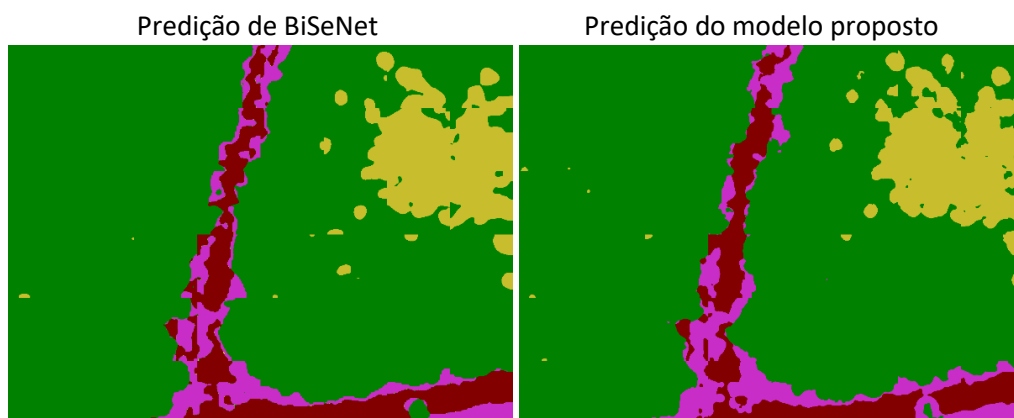
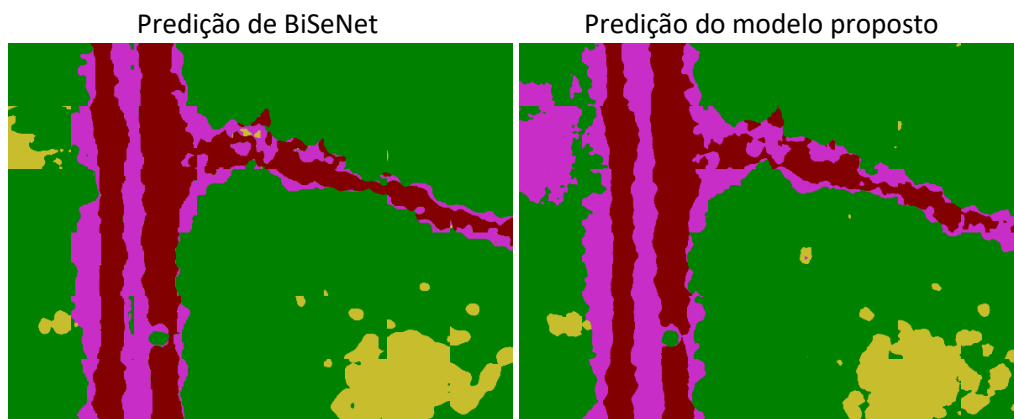
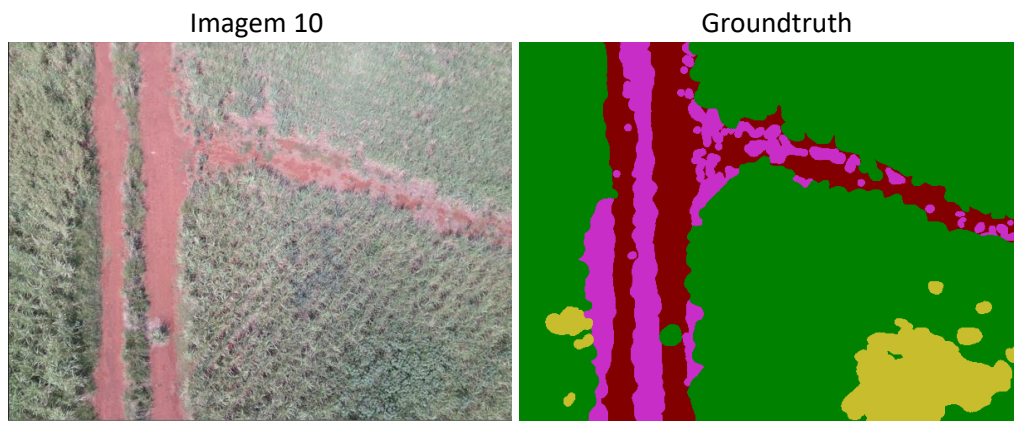
**Figura 7.5** – Comparação do modelo proposto com a rede BiSeNet.

### 7.1.2 Resultados qualitativos com imagens de teste RGB

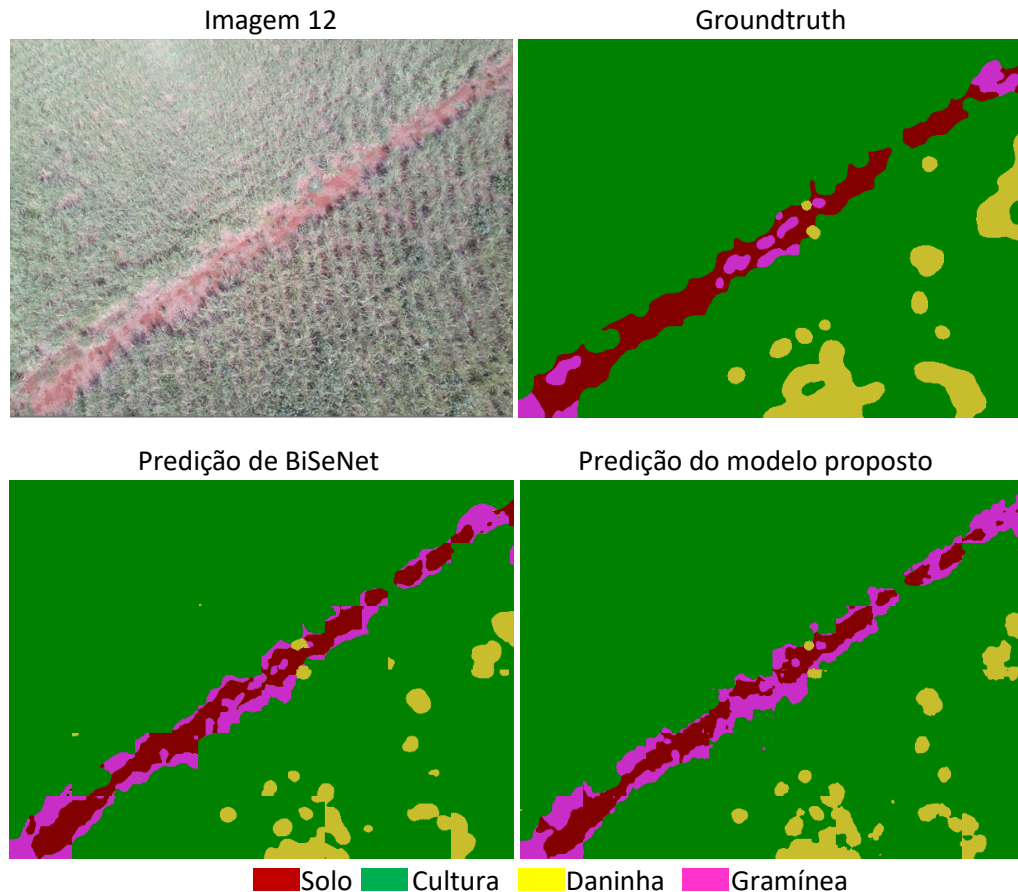
Para análise qualitativa do modelo proposto comparado ao modelo BiSeNet, treinados e testados com blocos de  $576 \times 256$  pixels (2 x GSD), apresentamos as três imagens aéreas RGB de teste com seus respectivos *groundtruths* e as imagens preditas por cada rede na [Figura 7.6](#). Como podemos observar, a diferença entre as predições dos modelos não é tão significativa visualmente, mas assim como a rede BiSeNet, nosso modelo foi capaz de discriminar entre plantas daninhas e cultura de cana-de-açúcar.

A [Figura 7.7](#) mostra uma comparação dos blocos preditos a partir das três imagens aéreas RGB de teste, gerados pela rede BiSeNet e pela rede proposta. Cada bloco $_{x,y(n)}$  é identificado pela posição  $(x, y)$  na grade de blocos da imagem  $n$  de teste original RGB.





(continua)

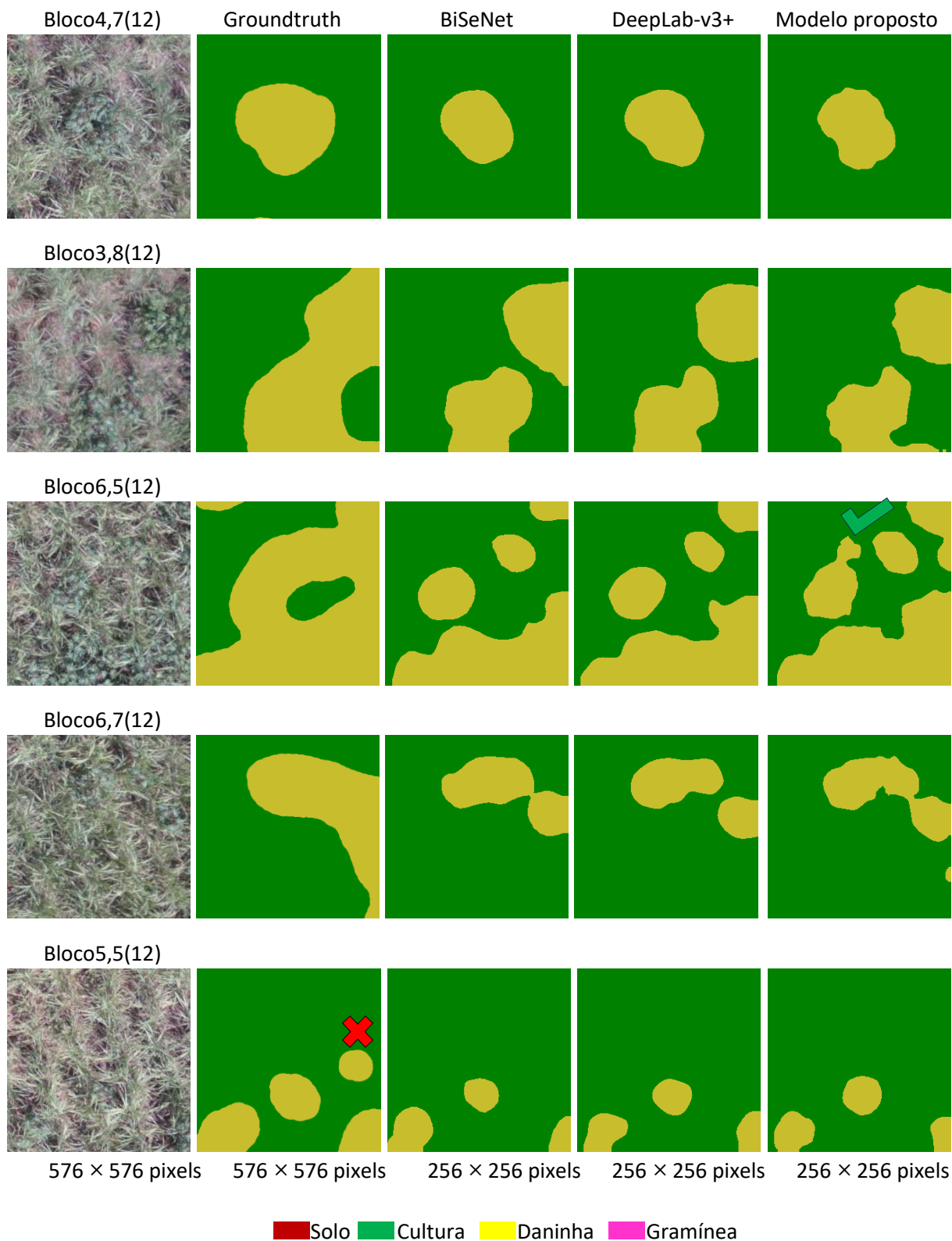


**Figura 7.6** – Comparação das previsões dos modelos de segmentação a partir de três imagens RGB de teste.

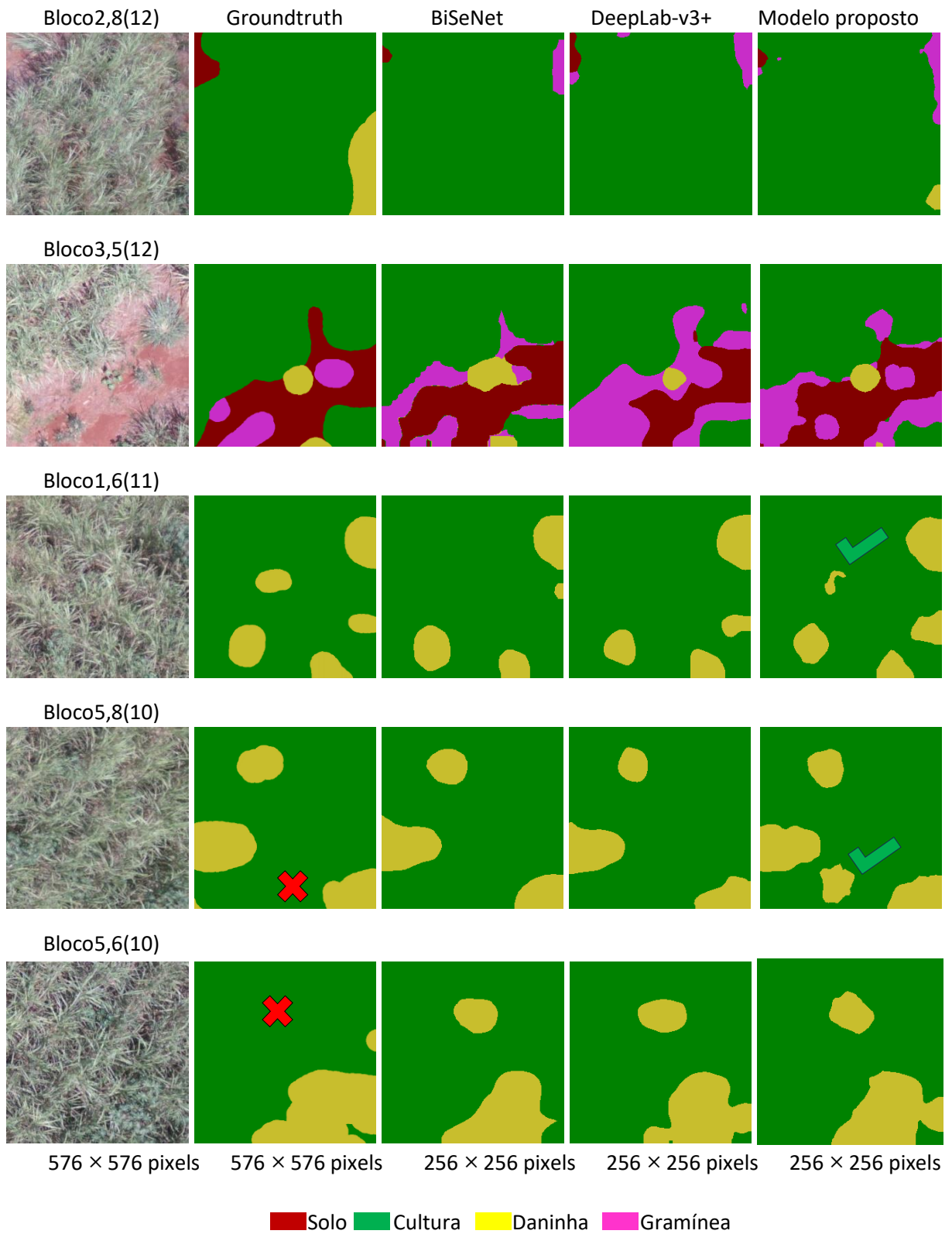
A classe daninha foi rotulada com bordas expandidas para evitar *pixels* não rotulados. Os modelos corrigem esta distorção, rotulando apenas as plantas daninhas sem bordas expandidas, como mostra o resultado do Bloco<sub>4,7(12)</sub> na Figura 7.7. O Bloco<sub>3,8(12)</sub> e Bloco<sub>6,5(12)</sub> mostram que os rótulos de plantas daninhas distintas foram unidos para simplificar a rotulação manual. No entanto, os modelos são capazes de identificar lacunas entre plantas daninhas. No Bloco<sub>6,7(12)</sub>, nosso modelo conseguiu separar corretamente as daninhas, embora exista uma união destas plantas no *groundtruth*.

No Bloco<sub>5,5(12)</sub>, houve um erro na rotulação manual, onde na verdade não existe visualmente a planta daninha, mas as redes rotularam corretamente. No Bloco<sub>2,8(12)</sub>, a daninha na borda do *groundtruth* não é visível no recorte do bloco original e não foi identificada pelas redes. No Bloco<sub>3,5(12)</sub>, uma planta daninha isolada foi corretamente classificada considerando as poucas amostras de daninha sobre solo. Percebe-se também a dificuldade de rotular manualmente os contornos e limites das classes solo e gramínea.

No Bloco<sub>1,6(11)</sub>, a daninha central tem difícil visualização pelo olho humano por estar coberta por folhas de cana-de-açúcar, tendo sido rotulada corretamente no *groundtruth* com auxílio da imagem multiespectral NIR ou *Red-Edge*. Nosso modelo consegue identificar plantas daninhas, mesmo nestes casos mais difíceis. No Bloco<sub>5,8(10)</sub>, a planta daninha central inferior é difícil de visualizar mesmo na imagem ampliada, mas foi corretamente classificada pelo nosso modelo. No Bloco<sub>5,6(10)</sub>, a daninha central visualmente perceptível não foi rotulada no *groundtruth*, mas foi corretamente predita nos modelos.



**Figura 7.7** – Blocos  $x, y (n)$  da imagem  $n$  de teste original RGB e seus groundtruths. Predições das redes BiSeNet, DeepLab-v3+ e modelo proposto, a partir de blocos da imagem original (continuação).



**Figura 7.7** – Blocos  $x, y (n)$  da imagem  $n$  de teste original RGB e seus groundtruths. Predições das redes BiSeNet, DeepLab-v3+ e modelo proposto, a partir de blocos da imagem original.

## 7.2 Resultados quantitativos com mosaico de teste RGB

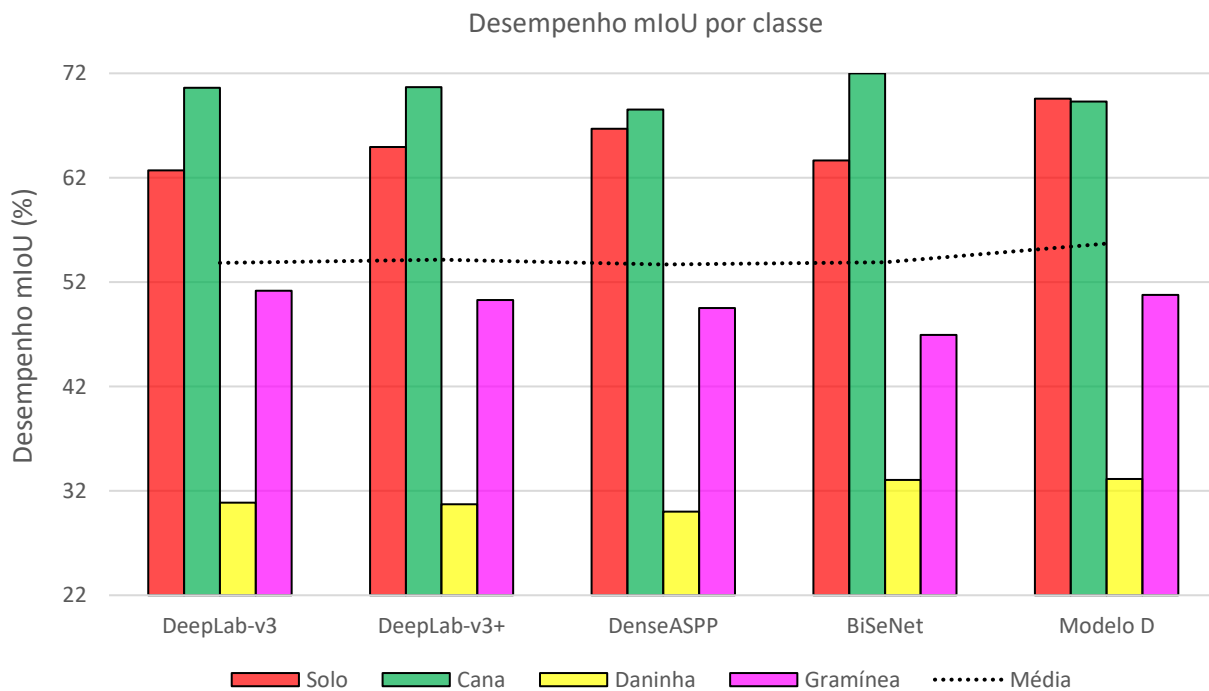
O modelo proposto – treinado com blocos de 576 » 256 (2,25 x GSD) das imagens aéreas RGB) – foi avaliado a partir do conjunto de blocos de 512 » 256 (2 x GSD) do ortomosaico de teste, sem os blocos inválidos. Os resultados quantitativos de IoU médio por classe são apresentados na [Tabela 7.7](#) e mostrados no gráfico da [Figura 7.8](#). O modelo proposto obteve valores semelhantes aos modelos selecionados, com melhor desempenho médio global. A média por classe e a média global são maiores quando comparadas com métodos mais complexos, como DeepLab-v3.

Apesar das distorções causadas pela ortorretificação do mosaico, o desempenho da classe daninha no mosaico foi até melhor do que o resultado apresentado a partir das três imagens originais de teste na [Tabela 7.3](#).

**Tabela 7.7** – Desempenho das melhores redes de segmentação semântica por classe no conjunto de teste composto por blocos do ortomosaico RGB.

Modelo de Segmentação	CNN-base	Solo %	Cana %	Daninha %	Gramínea %	Média do IoU	Configuração
DeepLab-v3	ResNet-50	62,69	70,61	30,88	<b>51,17</b>	53,84	AT(4)/DA
DeepLab-v3+	ResNet-50	64,93	70,68	30,71	50,27	54,15	AT(4)/DA
DenseASPP	ResNet-50	66,68	68,52	30,02	49,52	53,69	AT(3)/DA
BiSeNet	Xception	63,66	<b>71,98</b>	<b>33,04</b>	46,93	53,90	AT(5)/DA
Modelo D	VGG-16	<b>69,57</b>	69,29	<b>33,14</b>	<b>50,77</b>	<b>55,69</b>	AT(5)/DA

DA – Data Augmentation, AT – All Trainable

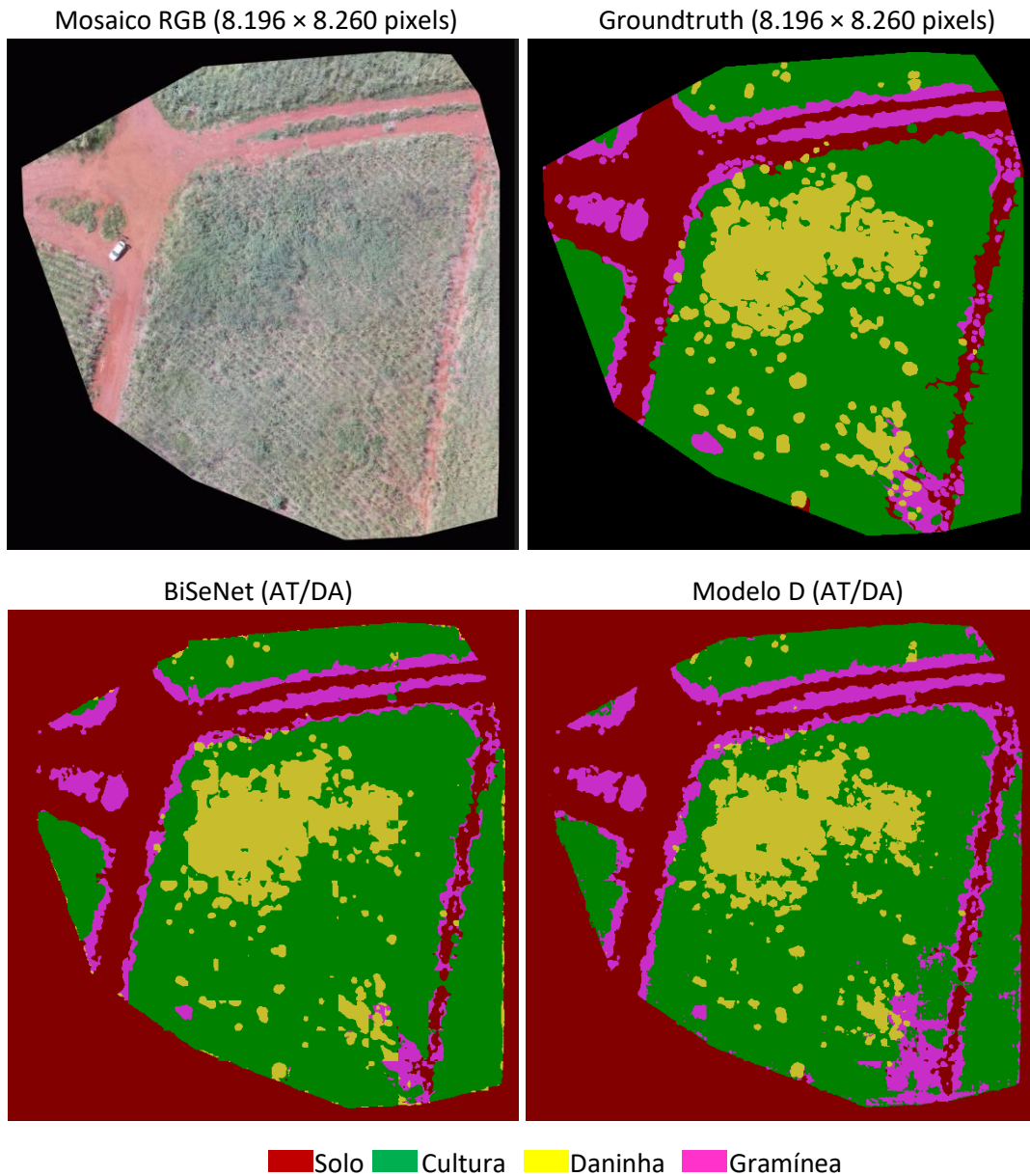


**Figura 7.8** – Desempenho IoU por classe dos modelos no conjunto de teste do mosaico.



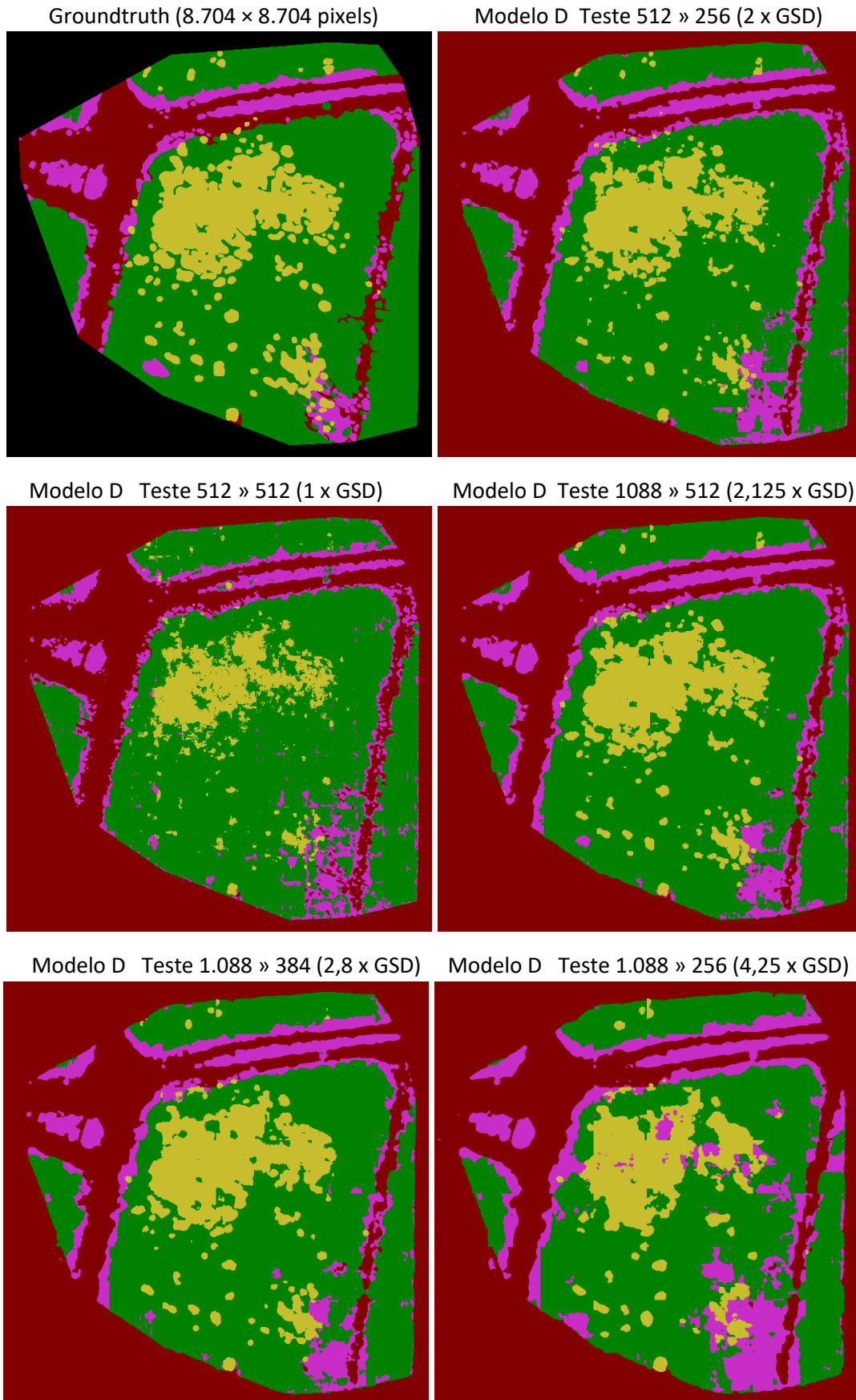
### 7.2.1 Resultados qualitativos com mosaico de teste RGB

A [Figura 7.9](#) mostra o resultado da predição do modelo proposto de segmentação semântica a partir dos blocos de  $512 \times 512$  pixels subamostrados para  $256 \times 256$  do ortomosaico RGB. A grade regular de  $17 \times 17$  blocos preditos foi recomposta para retornar ao tamanho original do mosaico subamostrado de  $4.352 \times 4.352$  pixels.

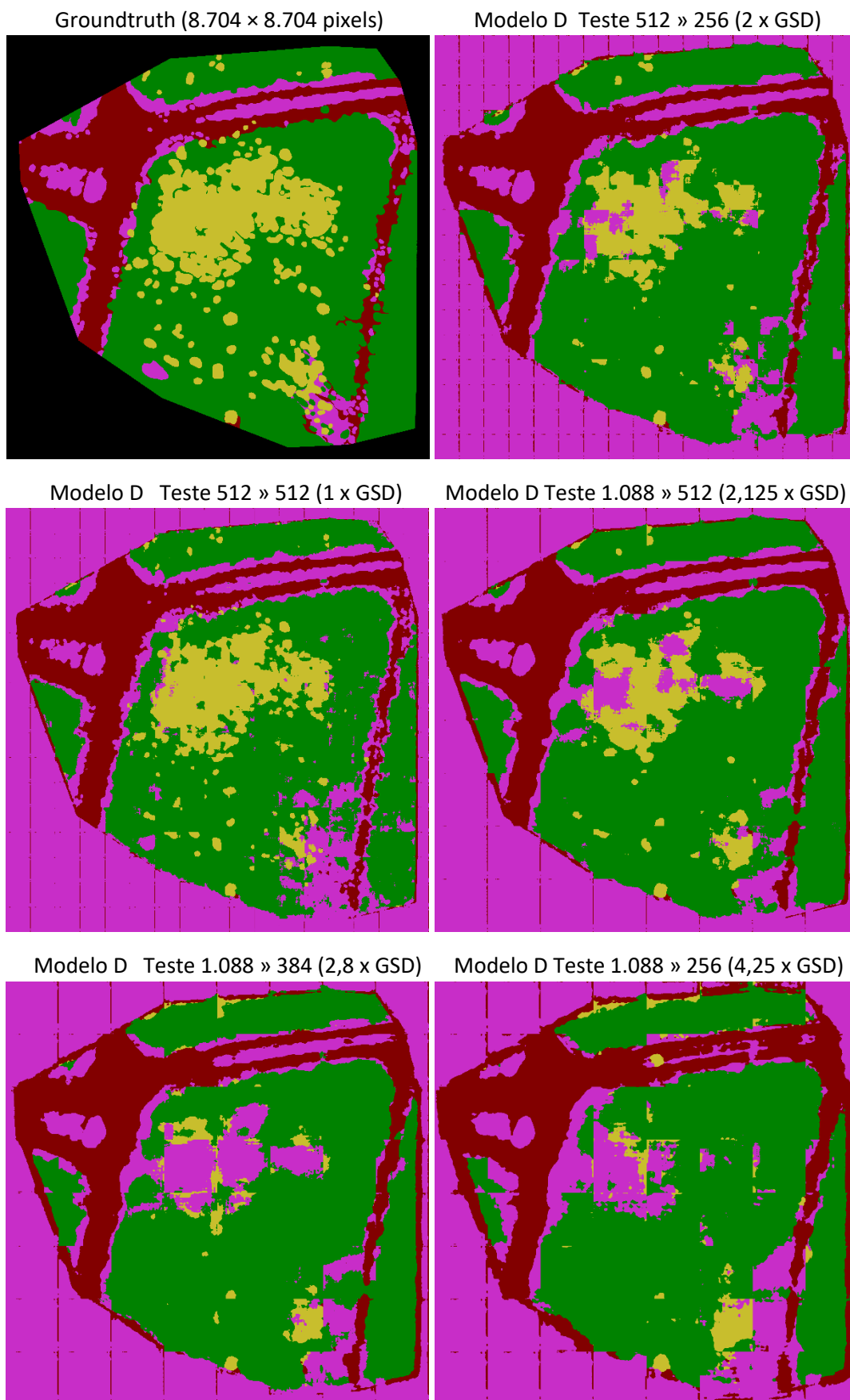


**Figura 7.9** – Predição do mosaico de tamanho  $4.352 \times 4.352$  pixels, sem máscara de borda, a partir dos blocos de teste  $512 \times 256$  do mosaico RGB, usando modelos de segmentação treinados com blocos de treinamento  $576 \times 256$  ( $2,25 \times$  GSD) das imagens aéreas RGB.

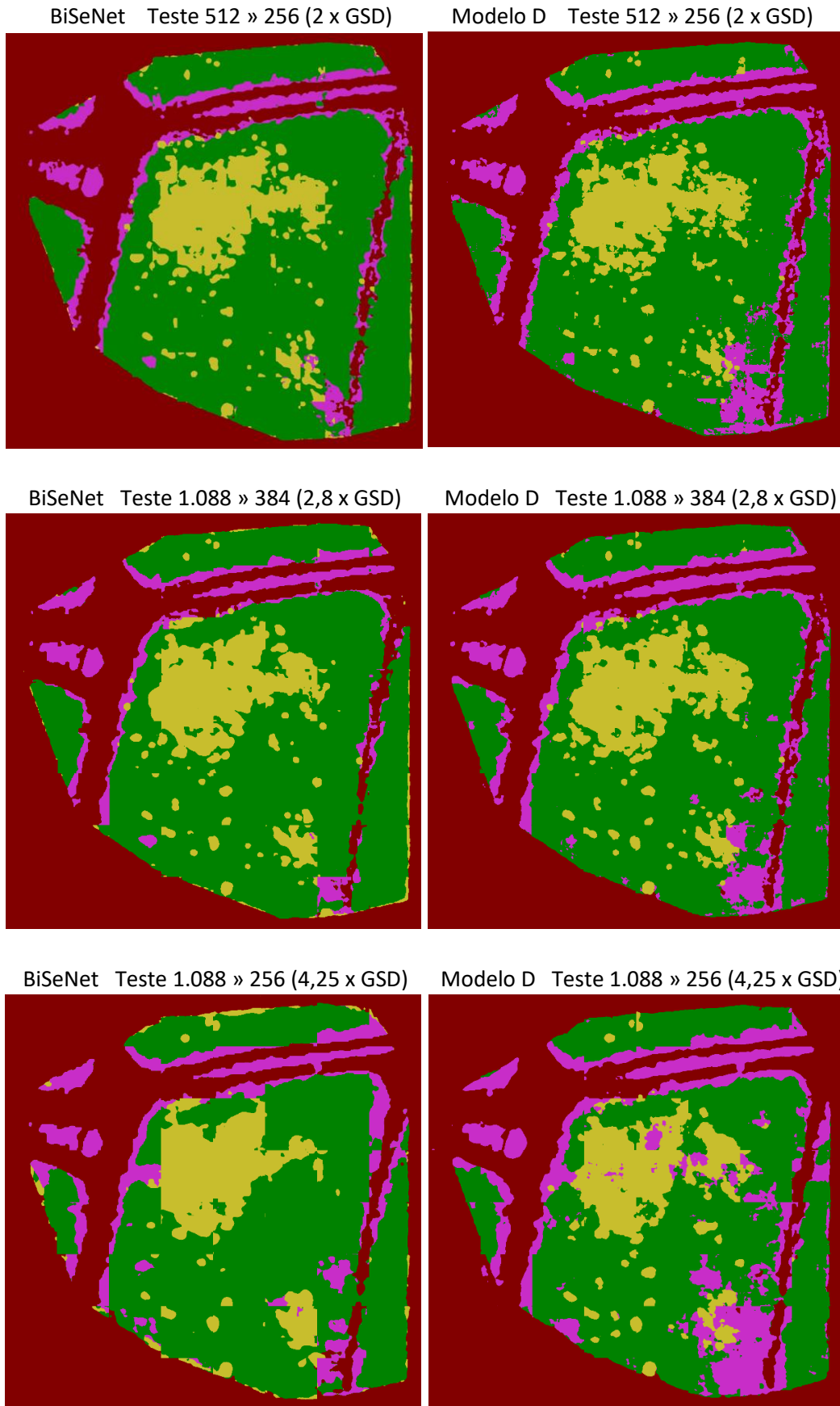
A [Figura 7.10](#) e [Figura 7.11](#) mostram os resultados da rede proposta treinada com  $576 \times 256$  ( $2,25 \times$  GSD) e  $576 \times 512$  ( $1,125 \times$  GSD), respectivamente, quando testadas com diferentes GSDs. A [Figura 7.12](#) e [Figura 7.13](#) mostram um melhor resultado da rede proposta em relação a rede BiSeNet.



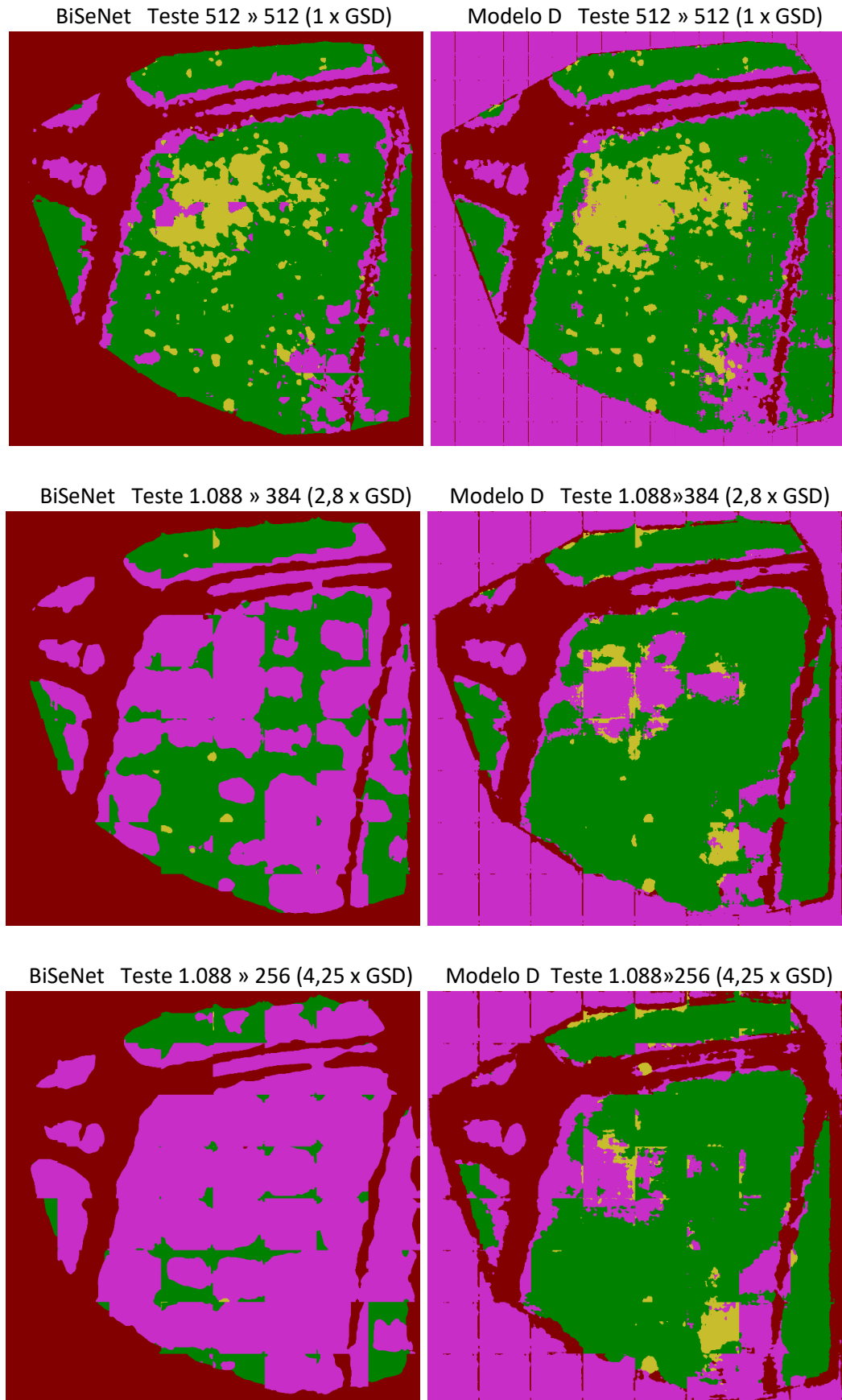
**Figura 7.10** – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelo proposto treinado com blocos de treinamento de tamanho 576 » 256 (2, 25 x GSD) das imagens aéreas RGB.



**Figura 7.11** – Predição do mosaico, sem máscara de borda, a partir dos blocos de teste de diferentes tamanhos e GSD do mosaico RGB, usando modelo proposto treinado com blocos de treinamento de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB.



**Figura 7.12** – Comparação da predição a partir de blocos de teste de 2,8 x GSD e 4,25 x GSD do mosaico RGB, usando modelo proposto e BiSeNet treinados com blocos de tamanho 576 » 256 (2,25 x GSD) das imagens aéreas RGB.



**Figura 7.13** – Comparação da predição a partir de blocos de teste de 2,8 x GSD e 4,25 x GSD do mosaico RGB, usando modelo proposto e BiSeNet treinados com blocos de tamanho 576 » 512 (1,125 x GSD) das imagens aéreas RGB.



### 7.3 Resultados com mosaico multiespectral multicanal

Devido ao desalinhamento das imagens multiespectrais originais, conforme discutido na [Seção 4.4.5.1](#), utilizamos diretamente os mosaicos de três canais multiespectrais *RedEdge–Green–NIR* para treinamento das redes, uma vez que os mapas de reflectância são alinhados por canal e calibrados radiometricamente pelo PIX4Dmapper. Assim, dividimos o mosaico multicanal em três partes para treinamento, validação e teste, como mostrado na [Figura A.8](#).

Adotamos a mesma abordagem de janela deslizante que opera em pequenas porções (*tiles*) do mosaico multiespectral. Definimos o tamanho do bloco como igual ao da entrada da rede de segmentação, sem subamostragem para evitar perda de resolução. Assim, para gerar o conjunto de dados na [Seção 5.1.5](#), dividimos o mosaico composto *RedEdge–Green–NIR* e seu *groundtruth* em uma grade de  $10 \times 12$  blocos de  $256 \times 256$  *pixels*, uma vez que este mosaico é relativamente pequeno. Blocos de  $512 \times 512$  *pixels* gerariam poucas amostras no conjunto de dados, com maior possibilidade de sobreajuste da rede.

Os blocos com fundo preto nas bordas do mosaico são descartados do conjunto de treinamento, como também dos conjuntos de validação e teste para efeito de medição de desempenho, uma vez que não definimos a classe “borda”. Desta forma, o conjunto de dados tem 44 blocos válidos de treinamento (dentro das áreas retangulares verdes), 16 blocos de validação (azul) e 15 blocos de teste (lilás) na [Figura A.8](#).

A [Tabela 7.8](#) e [Tabela 7.9](#) mostram o resultado quantitativo da rede BiSeNet e do modelo proposto. Nosso resultado foi superior à rede BiSeNet, que não detectou plantas daninhas em um mosaico de menor resolução (GSD de 4,63 cm/*pixel*). O mesmo problema aconteceu nos experimentos da [Seção 8.2](#). Mais experimentos devem ser realizados em trabalhos futuros para analisar melhor os resultados em mosaicos multicanais com composição de um número variável de mapas de reflectância e mapas de índices de vegetação.

**Tabela 7.8** – Desempenho IoU médio dos no conjunto de validação e teste composto por blocos do mosaico multiespectral *RedEdge–Green–NIR*.

Modelo de Segmentação	CNN-base	mIoU (%)		Configuração
		Val	Test	
BiSeNet	Xception	41,08	55,68	AT(5)/DA
Modelo proposto	VGG-16	67,81	83,38	AT(5)/DA

DA – Data Augmentation, AT – All Trainable

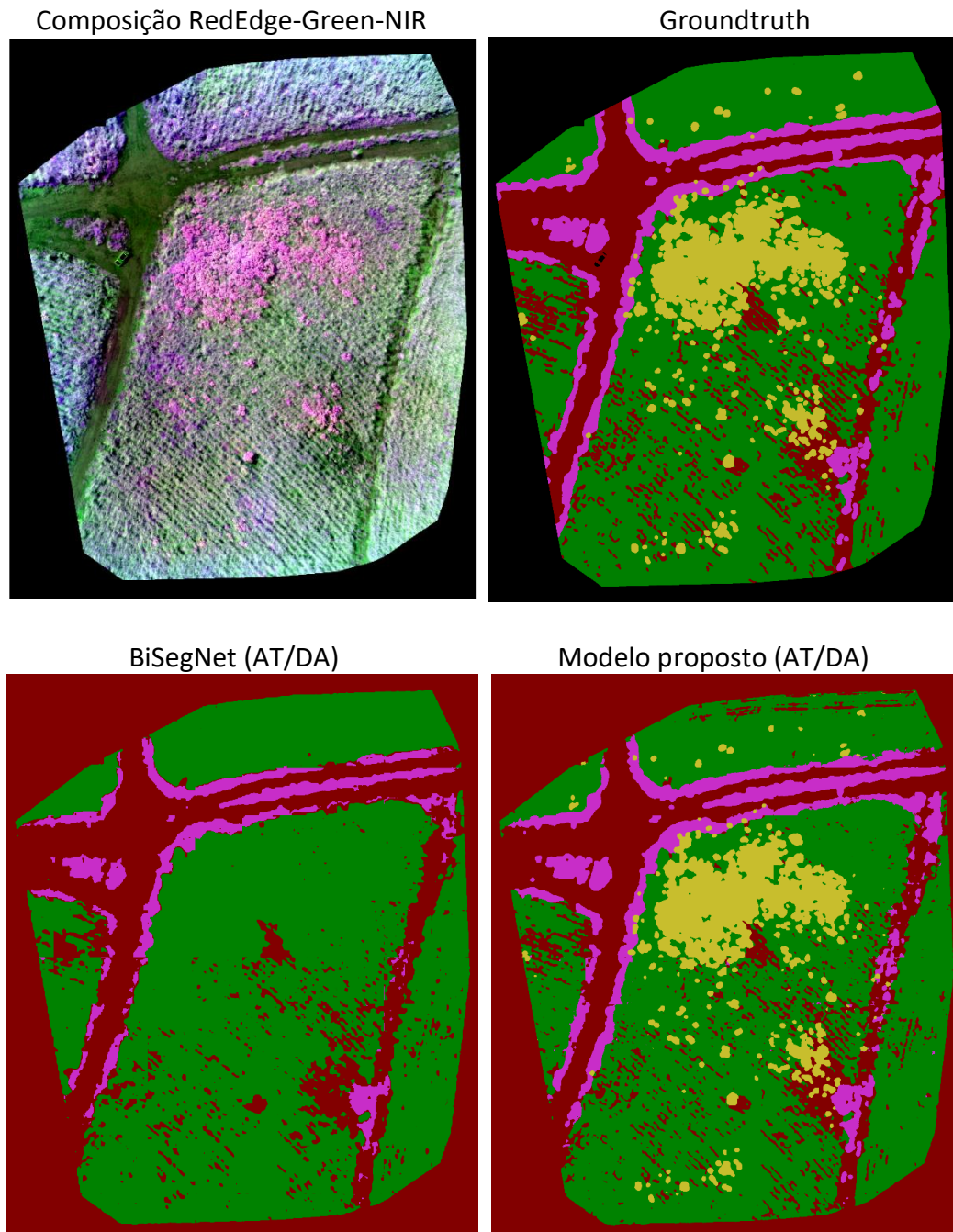
**Tabela 7.9** – Desempenho das redes de segmentação semântica por classe no conjunto de teste composto por blocos válidos do mosaico multiespectral.

Modelo de Segmentação	CNN-base (backbone)	Solo mIoU %	Cana mIoU %	Daninha mIoU %	Gramínea mIoU %	mIoU médio %	Configuração de rede
BiSeNet	Xception	41,80	68,45	00,00	24,50	33,69	AT(5)/DA
Modelo proposto	VGG-16	50,52	84,55	38,77	58,31	58,04	AT(5)/DA

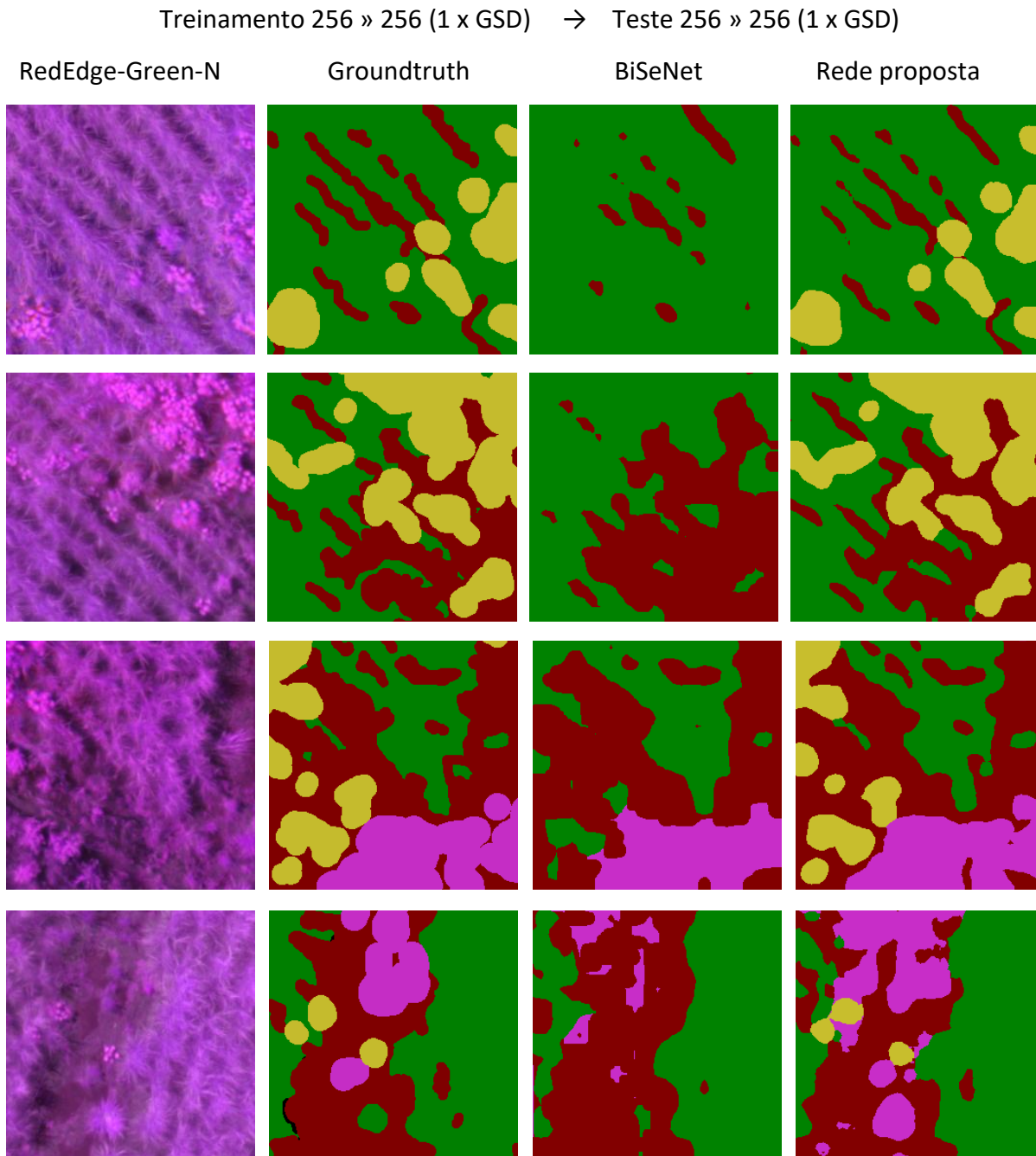
DA – Data Augmentation, AT – All Trainable

A predição completa do mosaico composto é realizada em todos os blocos válidos e inválidos, sendo necessário pós-processamento para recompor a grade de blocos e zerar a borda da predição com uma máscara de borda do mapa de reflectância.

A [Figura 7.14](#) apresenta os resultados qualitativos do mosaico composto pelas bandas *RedEdge–Green–NIR*, apenas para visualização da capacidade da rede proposta em detectar plantas daninhas em mapas multiespectrais, mesmo com um conjunto de dados tão pequeno e de menor resolução com GSD de 4,63 cm/pixel. Experimentos mais detalhados com uma maior quantidade de dados devem ser considerados em trabalhos futuros.



**Figura 7.14** – Predições dos blocos  $256 \times 256$  pixels sem subamostragem do mosaico *RedEdge–Green–NIR*, gerados pelas redes *BiSeNet* e modelo proposto, treinadas com blocos de  $256 \times 256$  pixels sem subamostragem.



**Figura 7.15** – Blocos preditos de  $256 \times 256$  pixels sem subamostragem do mosaico multiespectral composto RedEdge–Green–NIR, gerados pelas redes BiSeNet e rede proposta, treinadas com blocos de  $256 \times 256$  pixels sem subamostragem

A rede proposta obteve resultados visuais satisfatórios com a detecção precisa da classe daninha no mosaico multiespectral multicanal com GSD de  $4,63 \text{ cm/pixel}$ .

No próximo capítulo, apresentamos também alguns resultados visuais da segmentação semântica de plantas daninhas em um mosaico RGB com GSD de  $5 \text{ cm/pixel}$ , capturado com um VANT, em uma altura maior de voo, para mapear uma grande área de um campo de cultura de cana-de-açúcar, contendo plantas menores no início da fase de crescimento. Os resultados mostram que o modelo proposto supera a rede BiSeNet.



# Capítulo 8

## Avaliação em conjunto de dados extra

Neste capítulo, apresentamos apenas os resultados qualitativos do modelo proposto para segmentação semântica de plantas daninhas, em comparação com os modelos SegNet e BiSeNet, usando um ortomosaico RGB de imagens aéreas de menor resolução espacial capturadas por VANT, de dois talhões de cana-de-açúcar no início da fase de crescimento, para um manejo precoce de daninhas em uma aplicação real de agricultura de precisão.

### 8.1 Mosaicos RGB e *groundtruths*

O conjunto de dados extra é composto por dois ortomosaicos não georreferenciados (e sem as imagens aéreas originais) de dois talhões de uma cultura de cana-de-açúcar. Os mosaicos foram gerados com o software PIX4Dmapper a partir de imagens aéreas capturadas com uma câmera Canon G9X de 20,4 *megapixels* acoplada a um VANT de asa fixa da Horus Aeronaves, com altura de voo variando entre 125 a 200 metros para obter um mosaico de uma área maior do terreno, resultando em resolução espacial (GSD médio) de 5 cm/*pixel*. Cada ortomosaico foi rotulado manualmente por um biólogo nas classes solo, cana e daninha, usando o programa de manipulação de imagem GIMP [MONTEIRO 2019].

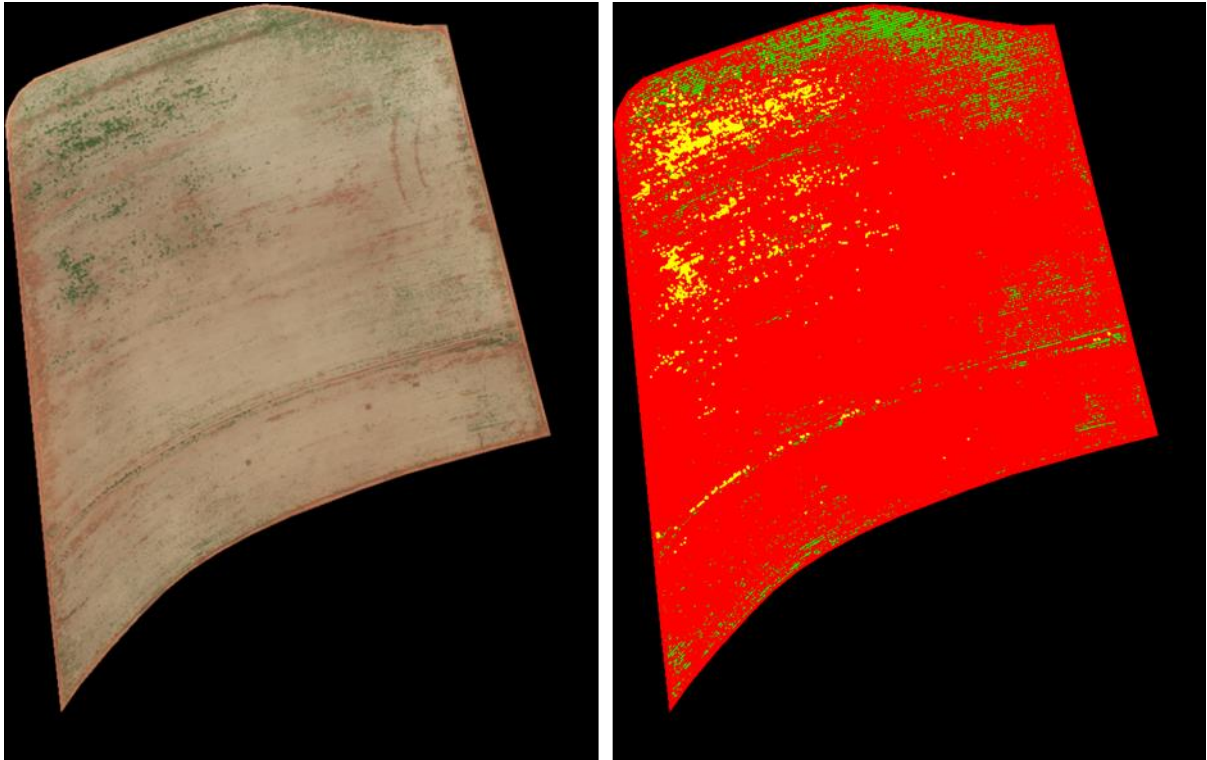
O ortomosaico *SugarcaneWeed* (com plantas daninhas) de  $5.364 \times 6.815$  *pixels* [MONTEIRO e WANGENHEIM 2019] e o ortomosaico *Sugarcane* (sem plantas daninhas) de  $9.391 \times 6.595$  *pixels* [PEREIRA JUNIOR e WANGENHEIM 2019] são disponibilizados *online* com seus respectivos *groundtruths* (Figura 8.1 e Figura 8.3, respectivamente). O mosaico sem plantas daninhas pode ser usado, por exemplo, em uma aplicação de segmentação de linhas de plantio para detecção de falhas no crescimento das mudas da cultura, mas neste trabalho utilizamos para aumentar o conjunto de dados de treinamento e validação com mais amostras da classe cultura de cana-de-açúcar, presentes em menor quantidade no mosaico *SugarcaneWeed*.

Para usarmos o mesmo código-fonte das redes de segmentação semântica testadas neste trabalho, os rótulos dos *groundtruths* originais codificados em cores RGB [solo: vermelho, cana: verde, daninhas: amarelo] foram transformados em rótulos de classe de 8 bits com os valores correspondentes [0, 1] em *Sugarcane* ou [0, 1, 2] em *SugarcaneWeed*.

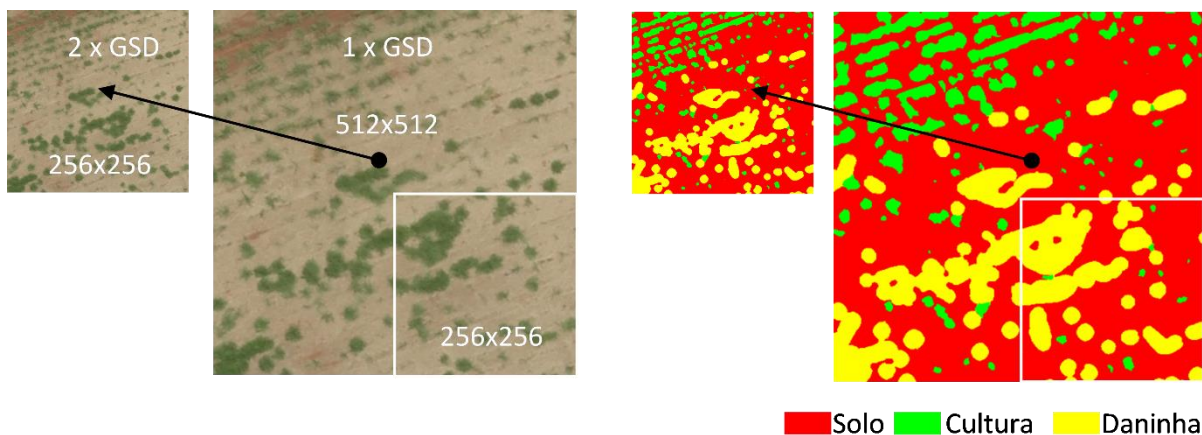
Devido ao grande tamanho dos mosaicos RGB, geramos o conjunto de dados das redes de segmentação, subdividindo os mosaicos e seus *groundtruths* em grades regulares de blocos de tamanhos menores. Para isso, primeiro expandimos ou cortamos as bordas pretas dos mosaicos e *groundtruths* para obter um número de linhas e colunas divisíveis por um número inteiro de blocos de tamanho 512 ou 256.



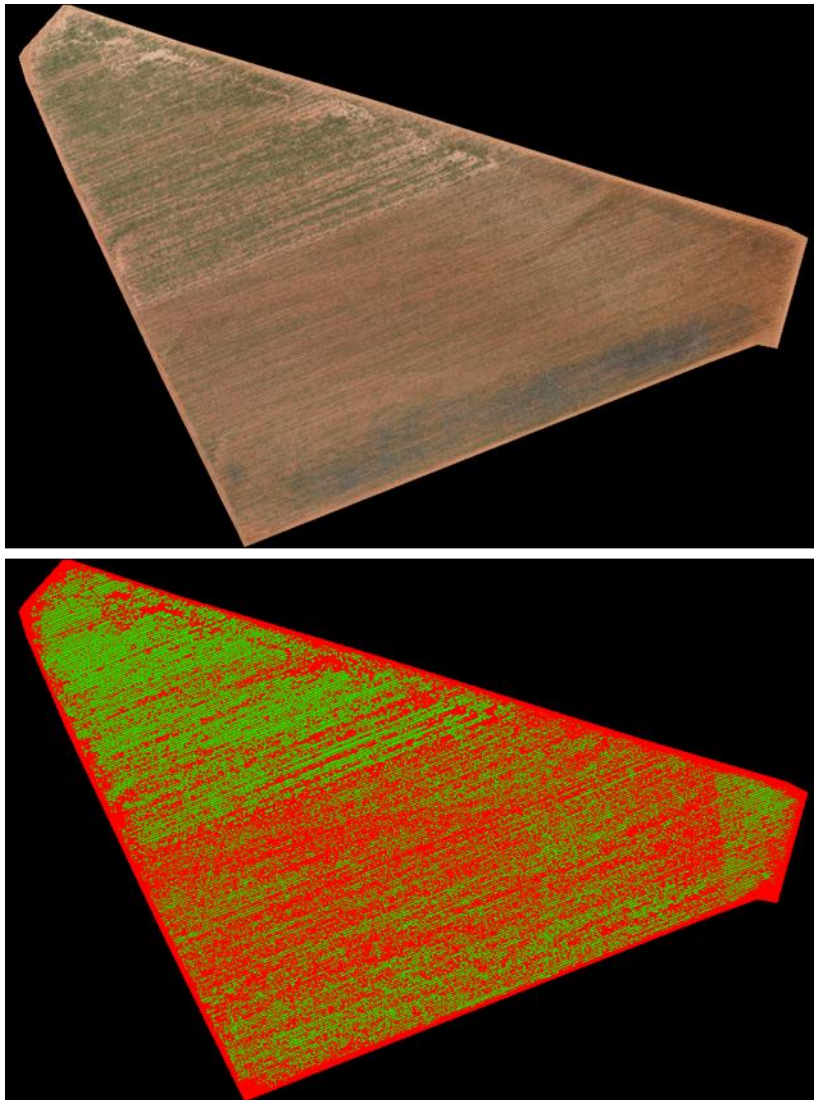
A [Figura 8.2](#) mostra detalhes de um dos blocos de *SugarcaneWeed*, de tamanho  $512 \times 512$  pixels ou  $256 \times 256$  pixels com resolução espacial de 1 x GSD (sem subamostragem), usados para treinamento, validação e teste das redes. Blocos subamostrados de 512 para 256 com 2 x GSD foram usados apenas para teste de desempenho com GSD diferente do treinado pela rede. A [Figura 8.4](#) mostra detalhes do mosaico *Sugarcane* dividido apenas em blocos de  $512 \times 512$  para treinamento da rede.



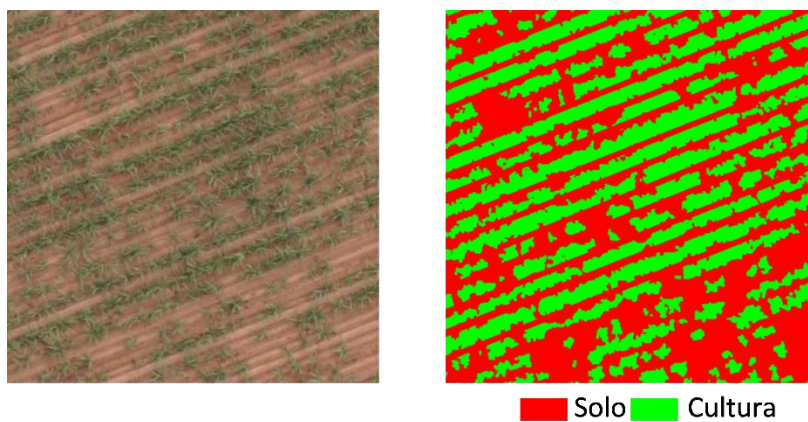
**Figura 8.1** – Ortomosaico RGB (*SugarcaneWeed*) e groundtruth com borda expandida para  $5.632 \times 7.168$  pixels com GSD médio aproximado de 5 cm/pixel [[MONTEIRO e WANGENHEIM 2019](#)].



**Figura 8.2** – Detalhes de um bloco com diferentes tamanhos e GSDs, com seus respectivos groundtruths com rótulos de classe solo, cultura de cana e daninha.



**Figura 8.3** – Ortomosaico RGB (sem plantas daninhas) e groundtruth cortados para  $9.216 \times 6.144$  pixels com GSD médio aproximado de 5 cm/pixel [PEREIRA JUNIOR e WANGENHEIM 2019].



**Figura 8.4** – Detalhes de um bloco de  $512 \times 512$  pixels, com maior quantidade de amostras de mudas de cana-de-açúcar, e seu groundtruth com rótulos de classe solo e cana.

## 8.2 Treinamento com blocos de $512 \times 512$ dos mosaicos RGB

Nesta seção, realizamos o treinamento das redes de segmentação semântica (SegNet, BiSeNet e rede proposta) usando os dois mosaicos *Sugarcane* e *SugarcaneWeed*. Os testes foram realizados apenas no mosaico *SugarcaneWeed* para simular uma aplicação de segmentação semântica de plantas daninhas.

Como utilizamos os dois mosaicos para treinamento das redes, há uma maior quantidade de blocos com instâncias da classe cana e poucas instâncias da classe daninha, como mostra a [Figura 8.1](#) e [Figura 8.3](#). Com isso, o desempenho das redes pode ser afetado com menor precisão na detecção da classe daninha.

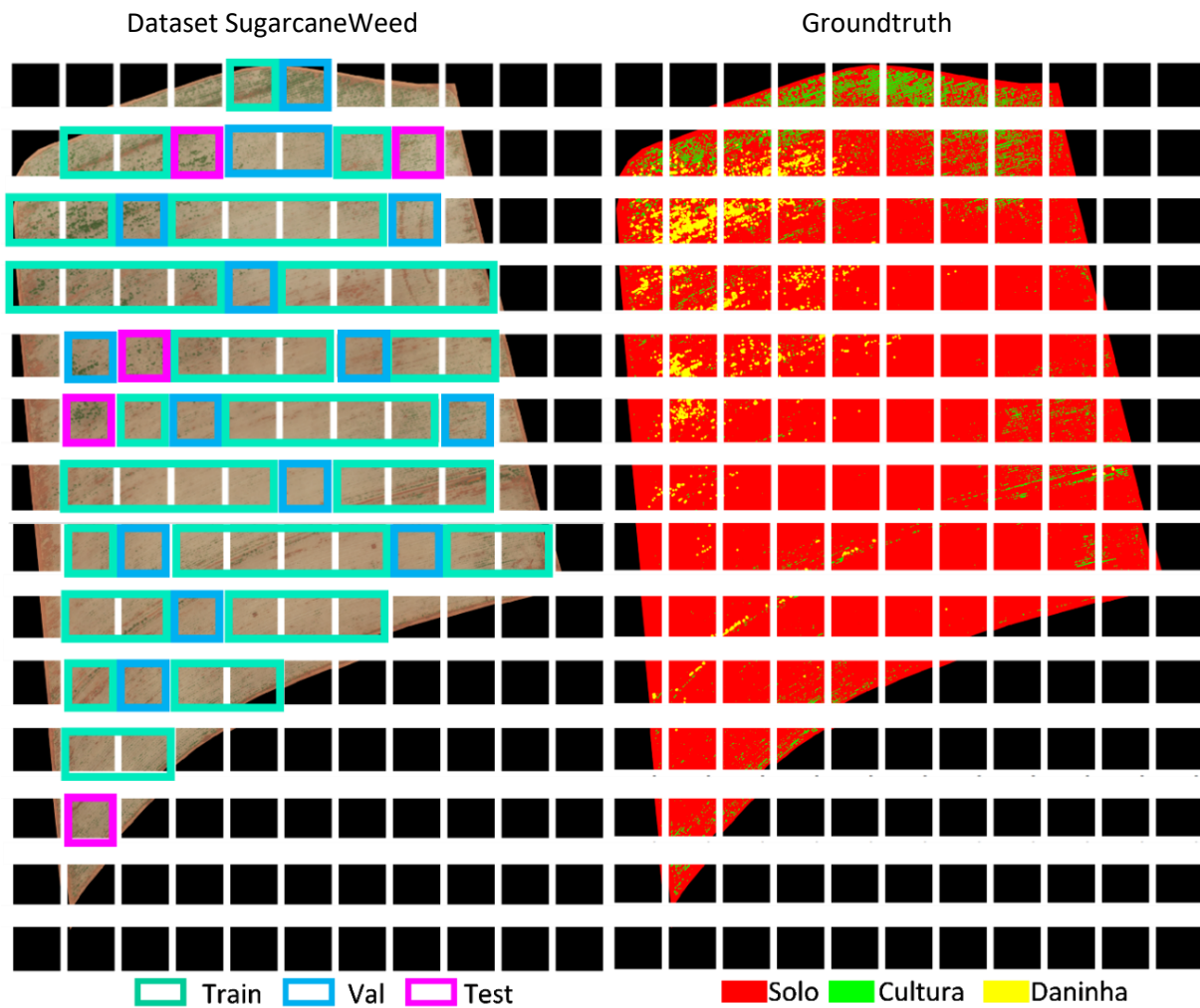
O mosaico *SugarcaneWeed* e seu *groundtruth* expandido de  $5.632 \times 7.168$  *pixels* foram subdivididos em uma grade regular de 11 x 14 blocos de  $512 \times 512$  *pixels*, mostrado na [Figura 8.5](#), enquanto o mosaico *Sugarcane* com *groundtruth* expandido de  $9.216 \times 6.144$  *pixels* foram divididos em 18 x 12 blocos de  $512 \times 512$  *pixels*, como mostra a [Figura 8.6](#). Os blocos inválidos contendo *pixels* pretos nas bordas dos mosaicos foram descartados do conjunto de dados de treinamento, validação e teste, uma vez que não definimos uma classe de fundo preto de pouco interesse neste tipo de aplicação.

Os blocos válidos de *SugarcaneWeed* foram selecionados manualmente em blocos de treinamento, validação e teste ([Figura 8.5](#)). Os blocos válidos de *Sugarcane* foram selecionados como blocos de treinamento ou validação ([Figura 8.6](#)). Desta forma, utilizamos um conjunto de dados de 160 blocos válidos, distribuídos em conjuntos de treinamento, validação e teste, com 123, 32 e 5 blocos, respectivamente.

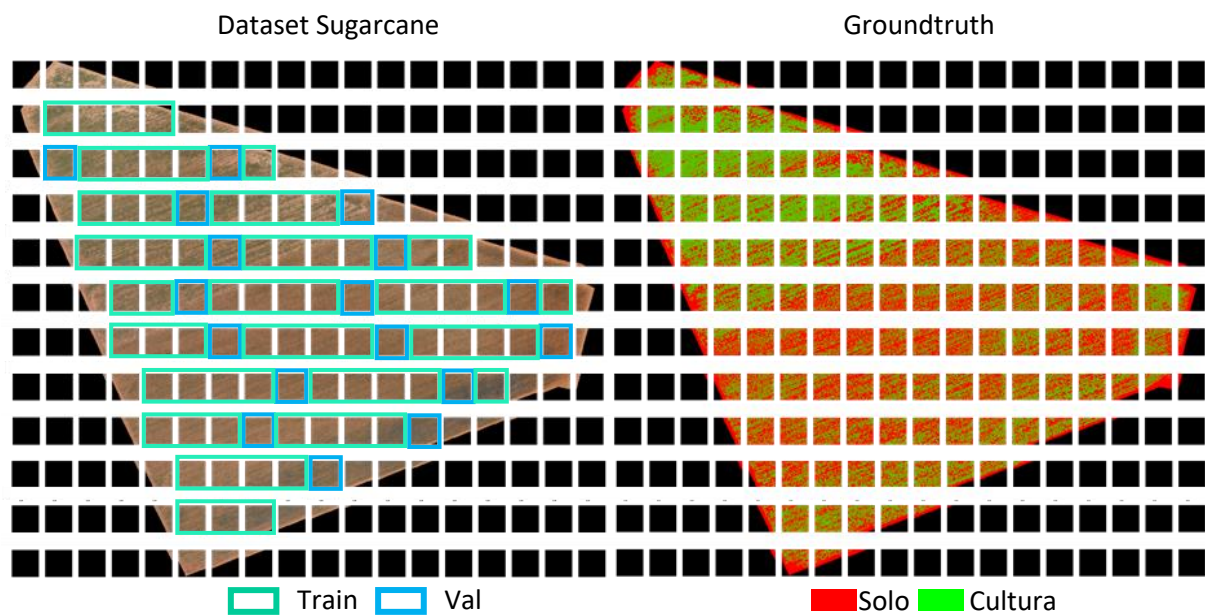
Neste experimento, as redes foram treinadas com blocos de  $512 \times 512$  *pixels* (com resolução espacial de 1 x GSD), sem subamostragem na entrada da rede, e testadas com dois GSDs diferentes. A [Seção 8.2.1](#) mostra os resultados qualitativos do teste realizado com blocos de  $512 \times 512$  *pixels* (1 x GSD), ou seja, mesmo tamanho e resolução utilizado no treinamento. A [Seção 8.2.2](#) mostra os resultados qualitativos do teste realizado com blocos subamostrados de  $512 \times 256$  *pixels* (2 x GSD), com resolução diferente do treinamento.

Os resultados qualitativos dos experimentos para comparação dos mapas de predição da rede proposta com as redes SegNet e BiSeNet são mostrados nas seções a seguir. Para visualização e análise qualitativa do mapa inteiro de predição, a predição foi executada no conjunto completo de dados do mosaico *SugarcaneWeed* (blocos válidos e inválidos), com as bordas do mapa de predição zeradas por uma máscara de borda extraída do mosaico RGB.





**Figura 8.5** – Ortomosaico RGB SugarcaneWeed e groundtruth de  $5.632 \times 7.168$  pixels divididos em uma grade regular de  $11 \times 14$  blocos de  $512 \times 512$  pixels.

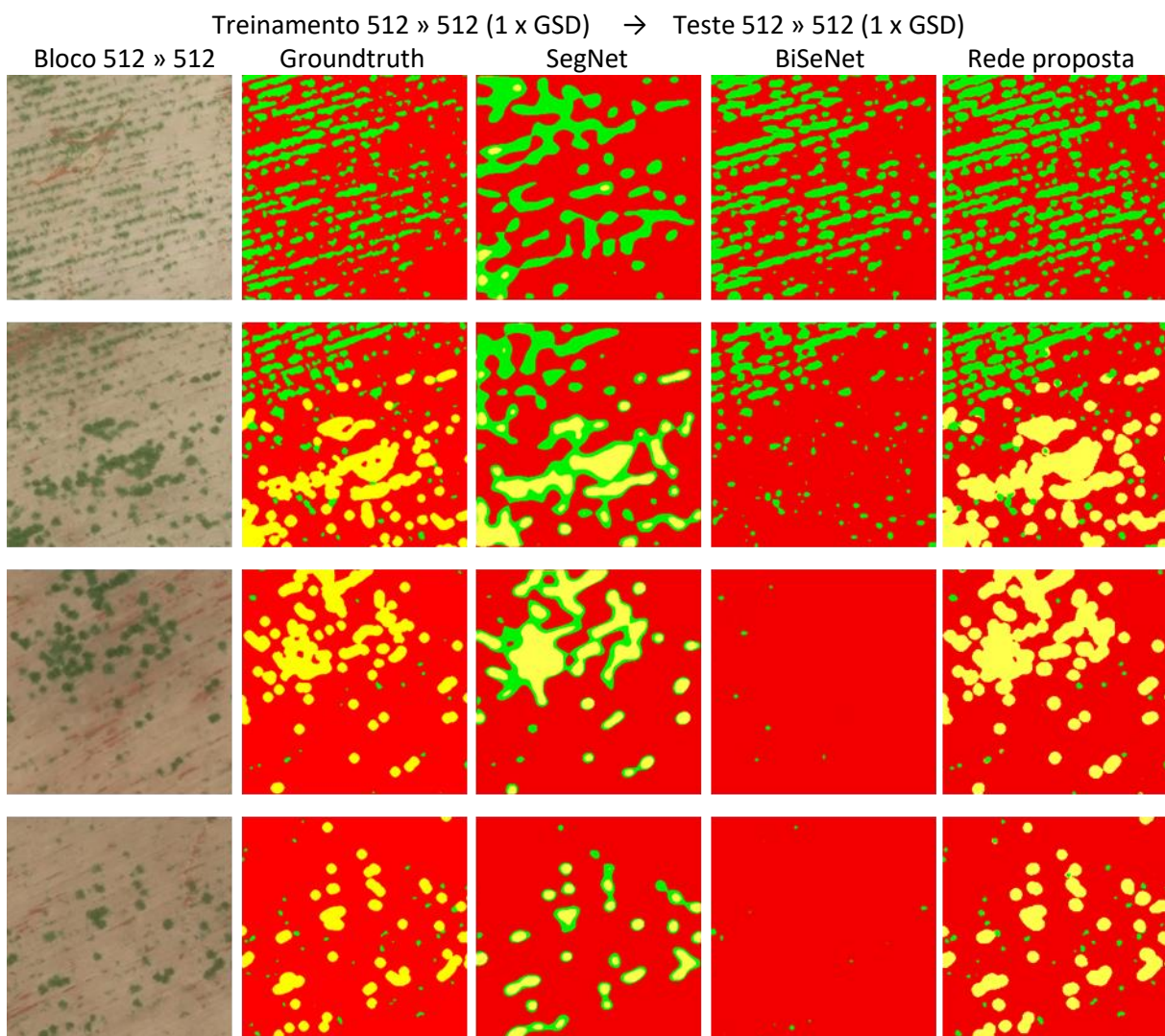


**Figura 8.6** – Ortomosaico RGB Sugarcane e groundtruth de  $9.216 \times 6.144$  pixels divididos em uma grade regular de  $18 \times 12$  blocos de  $512 \times 512$  pixels.

### 8.2.1 Resultado qualitativo de teste com bloco de 512 » 512 (1 x GSD)

Neste experimento, as redes foram treinadas com um conjunto de dados composto por blocos dos dois ortomosaicos (com e sem daninhas) com poucas amostras da classe daninha e predominância da classe solo e cana.

Na [Figura 8.7](#), a rede SegNet obteve um resultado qualitativo inferior ao apresentado por Monteiro [[MONTEIRO 2019](#)] devido às diferenças de implementação do código e estrutura da rede. Por exemplo, a rede SegNet com 34.968.004 parâmetros em [[MONTEIRO 2019](#)], que usa o código de referência implementado por [[SEIF 2019](#)], realiza *upsampling* com convolução transposta, enquanto a rede SegNet com 29.442.260 parâmetros utilizada neste trabalho, baseada na implementação de [[LU 2019](#)], faz uma interpolação bilinear menos precisa. Outras redes em nosso trabalho utilizam *backbones* diferentes de Monteiro e, por este motivo, comparações de desempenho não podem ser feitas diretamente sem uma análise mais detalhada.



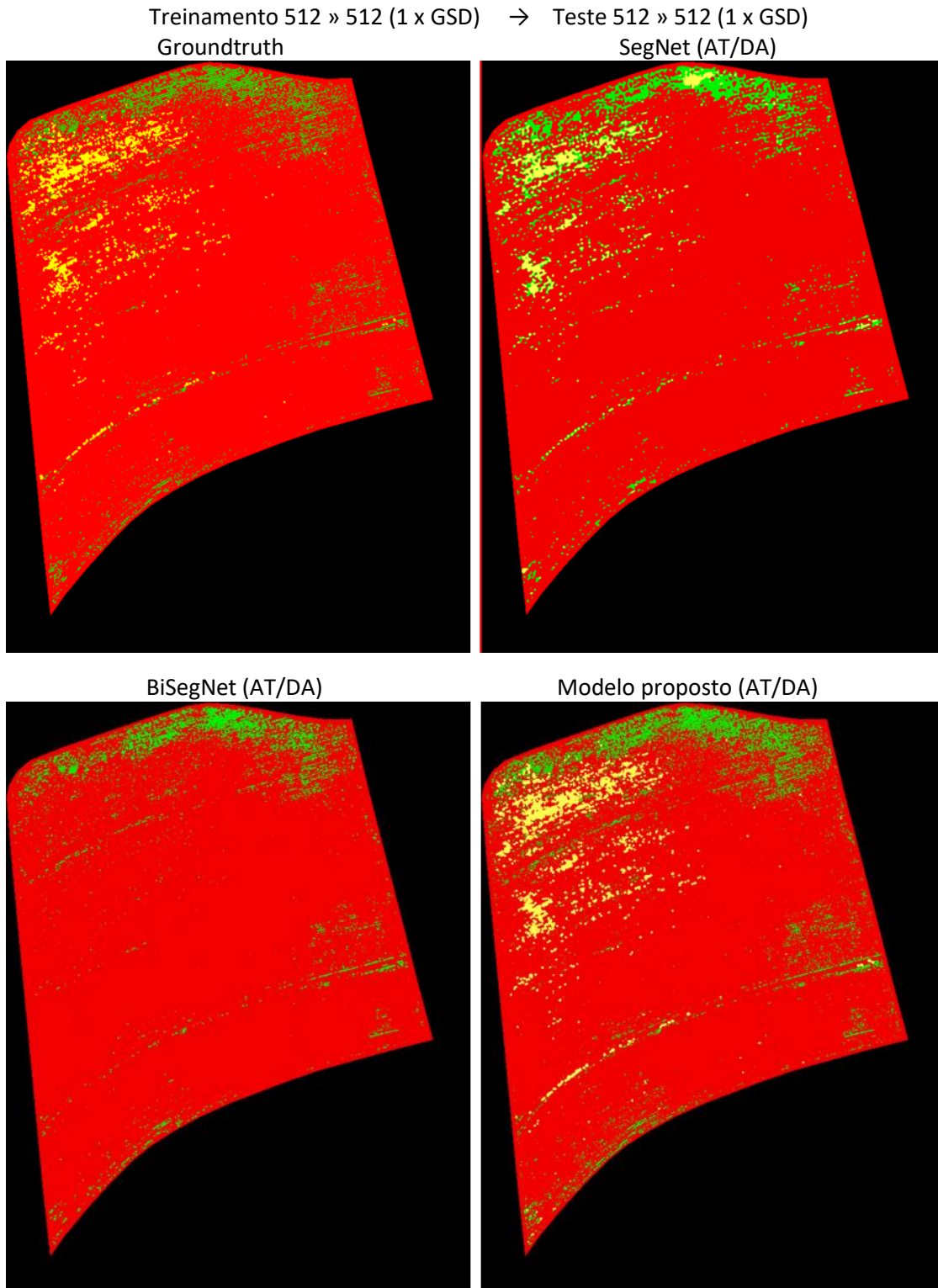
**Figura 8.7** – Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos 512 » 512 (1 x GSD) dos mosaicos RGB com GSD de 5 cm/pixel.

A [Figura 8.7](#) e [Figura 8.8](#) mostram que, em contraste com os bons resultados de teste obtidos pela rede BiSeNet no [Capítulo 5](#), esta rede não detectou nenhuma *pixel* da classe daninha, como também demonstrado em [[MONTEIRO 2019](#)], provavelmente devido



à combinação de menor quantidade de amostras desta classe, maior GSD e pequeno tamanho das plantas daninhas.

Por outro lado, a rede proposta obteve um bom desempenho de segmentação com limites de objetos mais refinados da classe cana, a partir de blocos de baixa resolução com GSD de 5 cm/pixel.



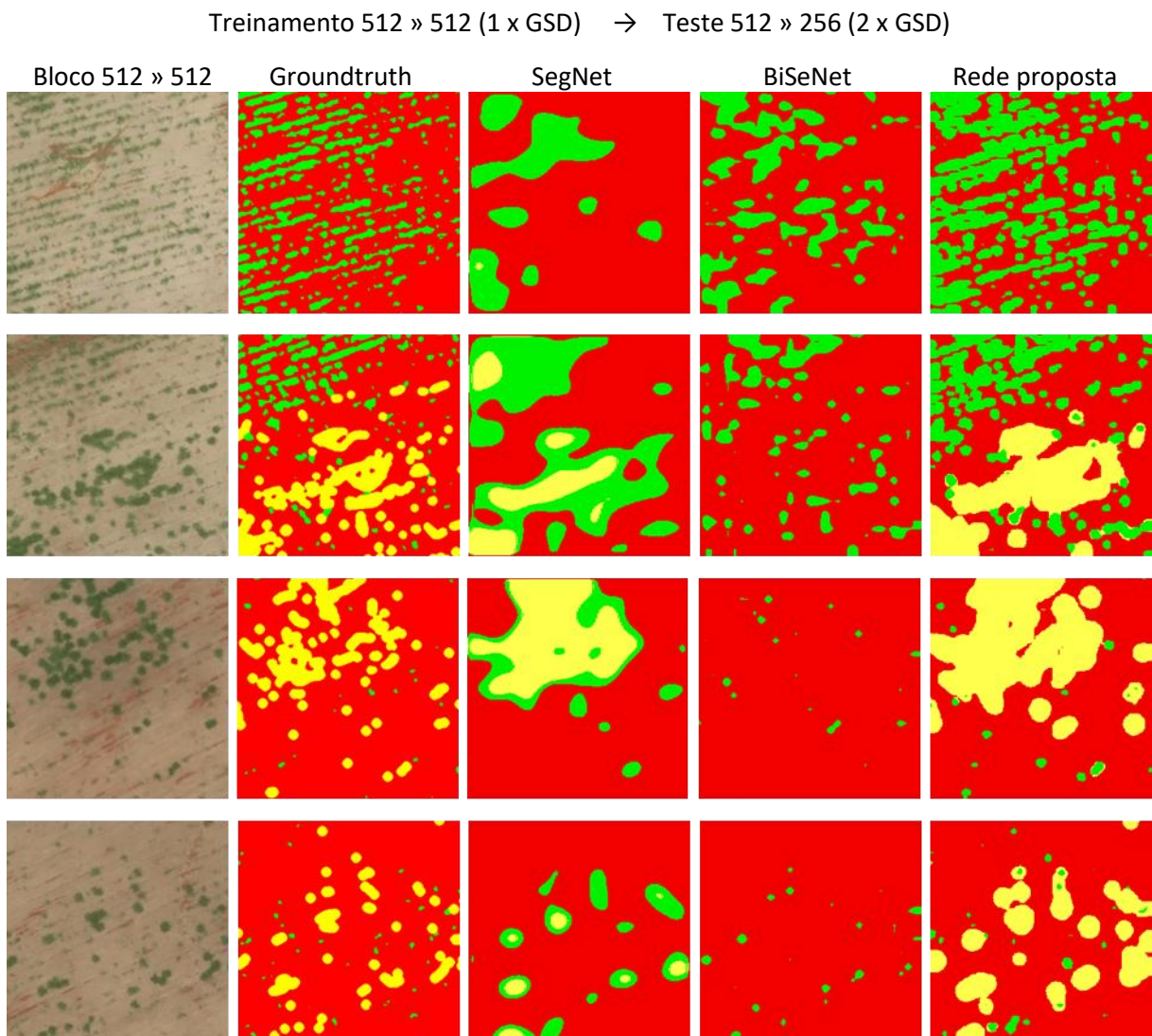
**Figura 8.8** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos 512 » 512 (1 x GSD) dos mosaicos RGB com GSD de 5 cm/pixel.

### 8.2.2 Resultado qualitativo de teste com bloco de 512 » 256 (2 x GSD)

Neste experimento, usamos blocos de teste com resolução espacial duas vezes menor que a resolução original de treinamento, subamostrando os blocos de 512 » 256 (2 x GSD), sendo o GSD médio do mosaico aproximadamente igual a 5 cm/pixel.

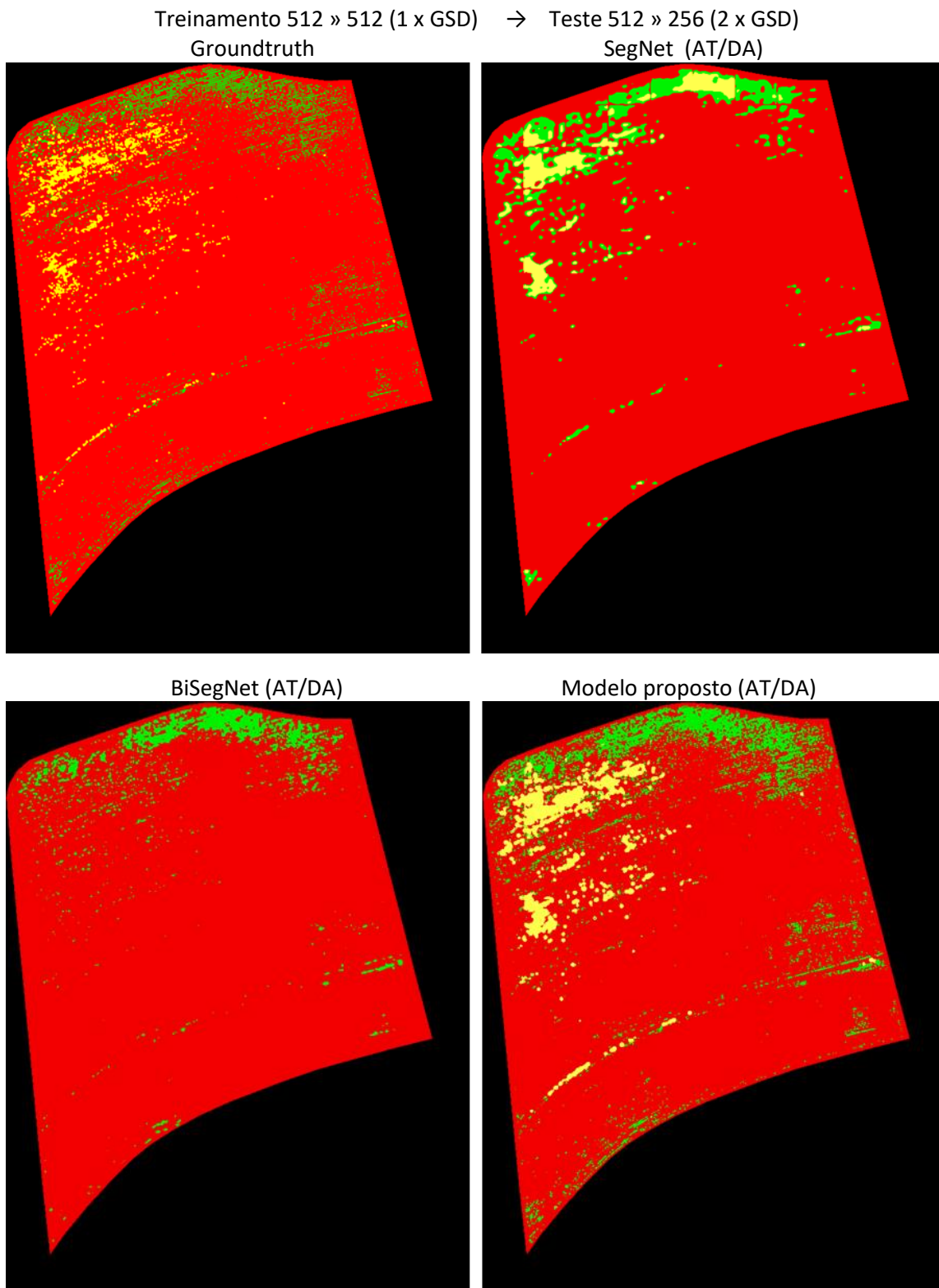
Na [Figura 8.9](#) e [Figura 8.10](#), a rede SegNet (sem atributos multiescala) apresentou uma segmentação semântica grosseira para as classes daninha e cultura, devido a perda de informações de detalhe, causada pela menor resolução espacial dos blocos de teste (10 cm/pixel) em relação a resolução dos blocos de treinamento (5 cm/pixel).

Na [Seção 3.4](#), a rede WeedMap [[SA et al. 2018](#)] também apresentou um desempenho razoável para previsão de *pixels* da classe cana, com um resultado ruim para daninhas, devido ao número de instâncias de plantas daninhas insuficientes no conjunto de dados de treinamento. Além disso, da mesma forma que nosso experimento, as imagens aéreas foram capturadas em fases iniciais de crescimento, com ervas daninhas muito pequenas para serem distinguidas nas imagens de baixa resolução.



**Figura 8.9** – Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento com blocos de 512 » 512 (1 x GSD) dos dois mosaicos RGB com GSD de 5 cm/pixel e teste com blocos de 512 » 256 (2 x GSD) do mosaico SugarcaneWeed.

No entanto, a rede proposta obteve um desempenho superior, com segmentação mais refinada na classe cana em comparação com a classe daninha, mesmo com blocos de teste de menor resolução com GSD correspondente a 10 cm/pixel.

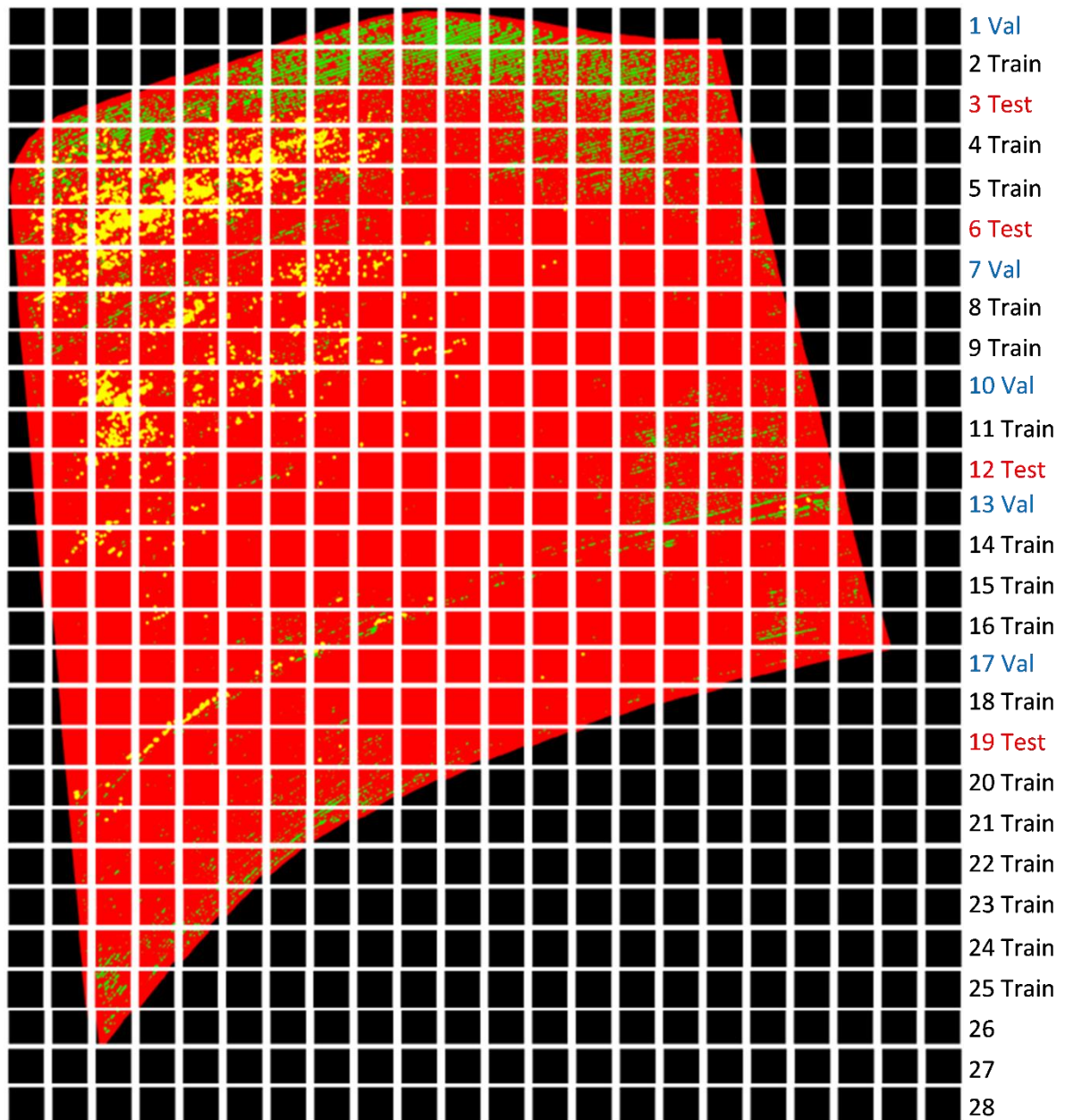


**Figura 8.10** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento com blocos de 512 » 512 (1 x GSD) dos dois mosaicos RGB com GSD de 5 cm/pixel e teste com blocos de 512 » 256 (2 x GSD) de SugarcaneWeed.



### 8.3 Treinamento com blocos de $256 \times 256$ dos mosaicos RGB

Nesta seção, as redes foram treinadas apenas com o conjunto de dados formado por blocos do ortomosaico *SugarcaneWeed*, ou seja, com pouca classe cana e predominância da classe solo e daninhas. Para isso, o mosaico e *groundtruth* correspondente de  $5.632 \times 7.168$  *pixels* foram divididos em uma grade regular de  $22 \times 28$  blocos de  $256 \times 256$  *pixels*, resultando em um conjunto de dados com 304 blocos válidos. Os blocos dos conjuntos de treinamento, validação e teste foram selecionados linha a linha na grade regular de blocos, com 182, 66 e 56 blocos, respectivamente (Figura 8.11).



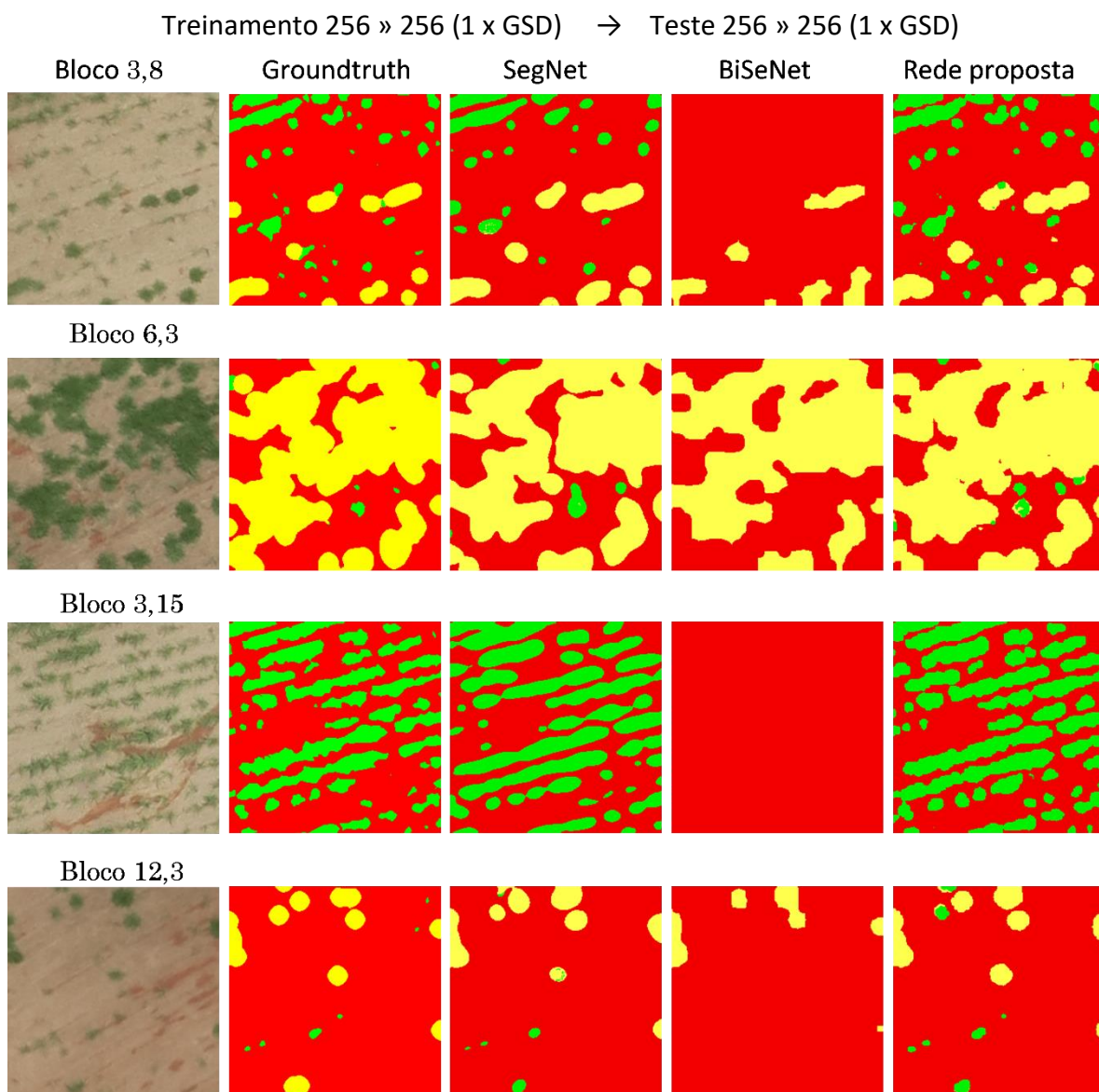
**Figura 8.11** – *Groundtruths* do ortomosaico *SugarcaneWeed* dividido em uma grade regular de  $22 \times 28$  blocos de  $256 \times 256$  *pixels*.

### 8.3.1 Resultado qualitativo de teste de 256 » 256 (1 x GSD)

Neste experimento, o treinamento das redes foi realizado apenas com blocos de 256 » 256 *pixels* (1 x GSD) do mosaico *SugarcaneWeed* e testado com blocos de mesmo tamanho e resolução.

Como não houve diferença de resolução do conjunto de teste em relação ao conjunto de treinamento, a rede SegNet obteve um bom resultado de segmentação.

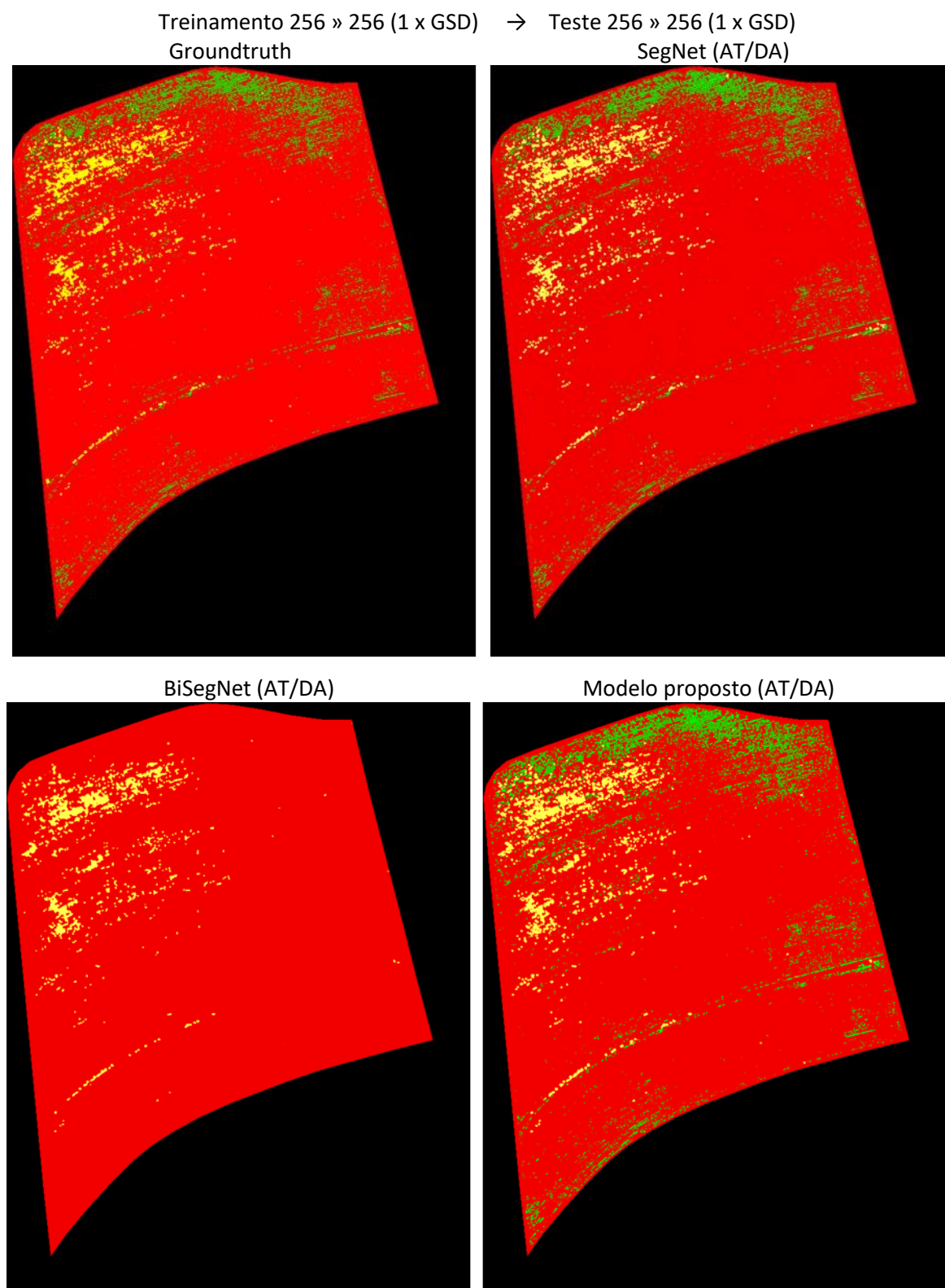
Ao contrário dos testes anteriores na [Seção 8.2](#), a rede BiSeNet detectou as plantas daninhas, mas não detectou nenhuma *pixel* da classe cana, provavelmente devido a pouca quantidade de amostras desta classe. Com base nos resultados destes experimentos, um balanceamento de classe, aumentando a quantidade de amostras da classe de maior interesse no conjunto de dados de treinamento deve melhorar os resultados obtidos.



**Figura 8.12** – Blocos de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos de 256 » 256 (1 x GSD) do mosaico *SugarcaneWeed* com GSD de 5 cm/*pixel*.



A [Figura 8.12](#) e [Figura 8.13](#) mostram que a rede proposta obteve um bom desempenho de segmentação semântica, mesmo com desbalanceamento de classes. No entanto, mais testes são necessários para avaliação da rede com GSD maiores.



**Figura 8.13** – Mapas de predições das redes SegNet, BiSeNet e rede proposta: treinamento e teste com blocos de 256 » 256 (1 x GSD) do mosaico SugarcaneWeed com GSD de 5 cm/pixel.

# Capítulo 9

## Conclusões

Após o mapeamento aéreo de uma cultura agrícola usando uma plataforma VANT equipada com sensores RGB e multiespectrais para geração de um ortomosaico georreferenciado de imagens de alta resolução espacial, técnicas de inteligência artificial usando redes neurais de aprendizado profundo podem ser aplicadas para mapeamento da distribuição de plantas daninhas, com o objetivo de diminuir o uso de produtos químicos para reduzir os impactos ambientais e aumentar a produtividade na agricultura.

Nesta tese, realizamos uma avaliação de desempenho quantitativo e qualitativo de redes de segmentação semântica baseadas em CNN para uso em aplicações de agricultura de precisão. A análise dos resultados indica que as redes multiescala, que utilizam estratégias para agregar informações de contexto em várias escalas para segmentação de objetos de diferentes tamanhos, obtêm melhores desempenhos de IoU médio, especialmente quando usadas em imagens de entrada com resolução espacial ou GSDs diferentes do utilizado no treinamento da rede.

Com base nas características dos modelos de melhor desempenho neste domínio específico, desenvolvemos uma arquitetura de rede de segmentação multiescala para detecção de plantas daninhas de diferentes tamanhos em cultura de cana-de-açúcar, a partir de mosaicos de alta resolução espacial, com escalas variáveis devido a variações de altura de voo ou escalas diferentes do conjunto de dados de treinamento da rede, sem necessidade de treinamento em diferentes escalas. A rede multiescala proposta obteve desempenho mIoU relativamente superior às redes de segmentação semântica já existentes, principalmente quando aplicada em mosaico aéreo de resolução espacial diferente do treinado pela rede.

Nas próximas seções, apresentamos uma discussão sobre os resultados de desempenho das redes de segmentação semântica em mosaicos RGB ou multiespectrais de imagens aéreas de VANT para mapeamento da distribuição de plantas daninhas em cultura de cana-de-açúcar, bem como as limitações desta pesquisa e direções para trabalhos futuros.

### 9.1 Discussão dos resultados

Para desenvolvimento deste trabalho, fizemos uma revisão bibliográfica de redes de segmentação semântica baseadas em CNN, conforme relatado no [Capítulo 2](#), e levamos em consideração os problemas e desafios levantados por trabalhos correlatos discutidos no [Capítulo 3](#), que usam redes neurais para segmentação semântica de mosaicos RGB e/ou

multiespectrais de VANTs em aplicações de detecção de plantas daninhas em diferentes culturas. Por exemplo, a rede WeedMap, baseada na rede de segmentação SegNet, obteve um desempenho razoável para previsão de cultura, mas um desempenho significativamente inferior para classificação de ervas daninhas. Isto ocorreu devido ao fato de as daninhas serem muito pequenas para serem distinguidas nas fases iniciais do plantio e haver um desbalanceamento de classes, com instâncias insuficientes de daninhas no conjunto de dados de treinamento. Observamos os mesmos problemas quando usamos o conjunto de dados de teste de um mosaico RGB extra na [Seção 8.2](#), onde a rede SegNet obteve um desempenho insatisfatório e a BiSeNet não foi capaz de detectar plantas daninhas. No entanto, argumentamos que isto ocorre principalmente com redes que não incorporam atributos multiescala suficientes para detectar objetos de tamanhos variados, nem preservam atributos de alta resolução com detalhes de objetos pequenos.

Para superar estas limitações, propomos uma rede multiescala para melhorar o desempenho de segmentação nesta área de aplicação específica. Para isso, realizamos um estudo e testamos nove modelos de redes neurais de aprendizado profundo, para segmentação semântica de plantas daninhas em cultura de cana-de-açúcar, usando imagens aéreas e mosaico RGB de VANT com variações de resolução espacial (GSD).

Primeiro, comparamos as estratégias de inicialização dos pesos da CNN-base – com pesos aleatórios ou por transferência de aprendizado com pesos pré-treinados no conjunto de dados ImageNet. Confirmamos que os pesos transferidos e ajustados usando o conjunto de dados específico geram melhores representações de atributos nas camadas da rede. Assim, treinamos as redes em nosso conjunto de imagens aéreas RGB (com transferência de aprendizado) e testamos no conjunto de teste de imagens ou mosaico RGB. Uma técnica de aumento artificial de dados para simular condições reais de voo, com rotações aleatórias, inversão horizontal e vertical, além de variações de iluminação, foi usada para melhorar o desempenho da segmentação e evitar o sobreajuste da rede. Com isso, todas as redes foram capazes de segmentar as plantas daninhas, apesar das distorções e perda de detalhes causadas pelo procedimento de ortorretificação do mosaico.

Em seguida, testamos as redes em diferentes resoluções espaciais. Em nossos experimentos, a maioria dos modelos mostrou bom desempenho na tarefa de segmentação, quando a resolução espacial de teste é a mesma utilizada no treinamento. No caso de imagens aéreas, isso significa que o GSD das imagens de teste (medido em  $cm/pixel$ ) deve ser o mesmo das imagens de treinamento, mantendo as mesmas condições de altura de voo e resolução da câmera. No entanto, quando o GSD de teste difere do originalmente treinado pela rede, os modelos multiescala apresentaram melhor desempenho do que outros modelos.

Com os resultados quantitativos e qualitativos de desempenho de todas as redes, analisamos as características dos nove modelos treinados e testados em nosso conjunto de dados. Com isso, conseguimos obter informações importantes para desenvolver novas redes de segmentação semântica multiescala baseadas em CNNs, como resumido a seguir.

### 9.1.1 Redes de segmentação semântica

As CNNs integram informações contextuais por meio de camadas sucessivas de subamostragem, que aumentam progressivamente o campo de visão, agregando o contexto com um campo receptivo cada vez maior até obter uma previsão de contexto global para

classificação de imagens. Porém, as operações de subamostragem descartam as informações de localização e detalhes de alta resolução. Por outro lado, os modelos de redes de segmentação semântica baseados em CNNs, requerem simultaneamente informação contextual de longo alcance e alta resolução para previsão densa em nível de *pixel*. Sendo assim, as redes de segmentação semântica baseadas em CNN utilizam alguma estratégia para recuperar a resolução perdida e, no caso de algumas aplicações com objetos de diferentes tamanhos, obter contexto multiescala.

A rede FCN extrai uma semântica de escala única dentro da imagem devido ao campo receptivo de tamanho fixo. Portanto, um objeto maior do que o campo receptivo pode ser fragmentado e mal rotulado, e um objeto menor do que o campo receptivo pode ser ignorado e classificado como fundo. Além disso, os detalhes de um objeto são perdidos ou suavizados devido ao procedimento simples de sobreamostragem do mapa de baixa resolução com fator 32 na saída da rede totalmente convolucional, empregado para recuperar a dimensão original da imagem de entrada.

A rede SegNet consiste em um codificador e decodificador correspondente para mapear os mapas de atributos de baixa resolução na saída do codificador para mapas de atributos com resolução original da entrada. O contexto espacial na camada mais profunda do codificador tem tamanho fixo e, portanto, a rede SegNet não agrega atributos multiescala. Além disso, as operações de deconvolução não são capazes de recuperar os atributos de alta resolução das camadas rasas que são perdidos após as operações de subamostragem no codificador.

A rede U-Net tem um codificador e decodificador com conexões de salto, que retêm informações espaciais dos atributos de camadas intermediárias para gerar previsão de alta resolução. As informações das camadas intermediárias são complementares aos atributos das camadas iniciais de convolução (com informações de baixo nível de abstração) e aos atributos de alto nível de camadas mais profundas (com informações semânticas de alto nível). Os atributos de todos os níveis são úteis para uma tarefa de segmentação semântica, uma vez que atributos semânticos de alto nível ajudam no reconhecimento de classes dentro de um contexto maior da imagem, enquanto atributos de baixo nível ajudam a gerar limites nítidos e detalhados para previsão de alta resolução de objetos pequenos. No entanto, a rede U-Net não agrega atributos com grandes campos receptivos, para obter um contexto maior em imagens de alta resolução.

Um módulo de contexto pode ser conectado aos modelos de segmentação existentes, para agregar informações contextuais sem perder resolução no topo da CNN. O módulo de contexto da rede DilatedNet possui apenas camadas convolucionais *atrous* em série (convoluções  $3 \times 3$  com taxas de dilatação [1, 1, 2, 4, 8, 16]), para agregar o contexto com campos receptivos crescentes. Isso permite ampliar o campo de visão (FoV) para incorporar um contexto maior na imagem de entrada da CNN, sem usar operações de subamostragem (mantendo a resolução espacial dos mapas de atributos na saída). No entanto, o módulo extra de convoluções *atrous* em série produz grandes campos receptivos de tamanho fixo e não captura atributos multiescala.

A rede PSPNet emprega uma pirâmide espacial com quatro camadas de *pooling* em paralelo para agregar informações multiescala com quatro tamanhos fixos de campo receptivo.

A rede DeepLab-v3 usa um módulo ASPP com quatro convoluções *atrous* paralelas com diferentes taxas de dilatação  $r = (6, 12, 18, 24)$  e, conseqüentemente, maiores campos de visão, muito úteis em imagens de alta resolução. Desta forma, a rede extrai informações multiescala, obtendo informações com diferentes campos receptivos em um mapa de atributos de entrada. No entanto, embora mais informações semânticas multiescala sejam codificadas no último mapa de atributos, informações detalhadas relacionadas aos limites do objeto são perdidas devido a subamostragens sucessivas da CNN, com redução da resolução da entrada por um fator de 16. Para amenizar este problema, a rede DeepLab-v3+ combina o módulo ASPP, capaz de codificar informações contextuais em várias escalas, com a arquitetura codificador-decodificador com conexões de salto de mapas de atributos de alta resolução para refinar os resultados da segmentação.

Embora o ASPP seja capaz de codificar atributos multiescala, o número de escalas é insuficiente para algumas aplicações. Além disso, quando as imagens são de alta resolução, o campo receptivo deve ser aumentado, exigindo uma taxa de dilatação maior. No entanto, conforme a taxa de dilatação aumenta com filtro mais esparsos e o campo receptivo se torna maior que o mapa de atributos de entrada, a convolução *atrous* torna-se cada vez mais ineficaz com a deterioração do *kernel*, que impede a captura de um contexto maior.

Para contornar estas limitações, a rede DenseASPP usa um módulo de convoluções *atrous* em série e paralelo, capaz de simultaneamente codificar informações em várias escalas de forma densa e atingir um tamanho suficientemente grande de campo receptivo (para capturar informações de maior contexto em imagens de alta resolução), sem aumentar a taxa de dilatação, evitando a deterioração do *kernel* de convolução *atrous*.

Apesar desses métodos indicarem que a informação espacial de alta resolução e campos receptivos maiores são cruciais para alcançar alta precisão, é difícil atender estes requisitos simultaneamente. Para resolver este problema, uma rede de segmentação bilateral (BiSeNet), contém duas partes: um caminho espacial para preservar as informações espaciais da imagem original, contendo um modelo leve com apenas três camadas de convolução; e um caminho de contexto com uma CNN leve para reduzir rapidamente o mapa de atributos e obter um grande campo receptivo, que codifica informações de contexto semântico de alto nível.

Na rede Hypercolumns, os mapas de atributos intermediários da CNN são reutilizados em ramos paralelos com conexões de salto para preservar a resolução de cada camada. Em seguida, os mapas são interpolados bilinearmente para uma maior resolução e concatenados em mapas de atributos multiescala. As respostas dos mapas de atributos nas camadas superiores codificam o contexto de longo alcance (contexto global), enquanto as camadas rasas preservam os detalhes de pequenos objetos (contexto local). Ainda assim, os mapas de atributos contêm poucas escalas agregadas.

Os resultados de desempenho dos testes das redes multiescala, como BiSeNet, DeepLab-v3 e DenseASPP, apresentaram melhores resultados no [Capítulo 5](#), principalmente quando analisados em blocos de teste de resoluções espaciais diferentes do treinamento. No entanto, a rede BiSeNet incorpora poucas escalas que pode não funcionar bem para imagens aéreas capturadas por VANT com GSDs maiores, em alturas de voo mais elevadas, quando as plantas daninhas são pequenas no início da fase de crescimento, como pode ser verificado nos resultados do [Capítulo 8](#). Porém, mais testes são necessários para obter maiores esclarecimentos.



### 9.1.2 Modelo de rede proposta

Com estas considerações, no [Capítulo 6](#), desenvolvemos um modelo de rede neural multiescala baseado em CNN, com um bloco denso de convoluções dilatadas, projetado especificamente para aplicações de agricultura de precisão, com objetivo de melhorar o desempenho da segmentação semântica de mosaicos de imagens aéreas de alta resolução de VANT, para mapeamento da distribuição de plantas daninhas em culturas agrícolas. Para atender aos requisitos desta aplicação, o modelo deve preservar as informações dos mapas de atributos de alta resolução da CNN e, simultaneamente, obter mapas de atributos com informações de contexto multiescala.

O bloco denso analisa os mapas de atributos intermediários da CNN com convoluções *atrous* em série e em paralelo, com diferentes combinações de uma a quatro taxas de dilatação  $r = [3, 6, 12, 18]$  e, conseqüentemente, diferentes campos receptivos. Com as convoluções *atrous* em série, é possível analisar o mapa de atributos de alta resolução da CNN com um campo de visão máximo para obter maior contexto, sem deteriorar o *kernel*, e simultaneamente, analisar esta entrada em escalas menores de campos receptivos concatenando as saídas do bloco denso, o que corresponde à convoluções *atrous* em paralelo. Assim, é possível obter atributos multiescala com um número maior de escalas. No modelo proposto, a saída do bloco denso é concatenada com conexões de salto dos mapas intermediários da CNN, combinando informações semânticas em várias escalas para refinar a previsão de alta resolução.

Em nossos experimentos, conseguimos aumentar o desempenho da rede proposta a partir de um mosaico de alta resolução de VANT, quando a entrada de teste tem um GSD diferente do originalmente treinado pela rede. O modelo proposto atingiu melhores resultados multiescala em comparação com modelos de segmentação mais complexos ou profundos. Com isso, concluímos que os mapas de atributos de alta resolução das camadas intermediárias da CNN fornecem informações úteis detalhadas para segmentação das imagens, e que o bloco denso (paralelo à CNN) é capaz de recuperar informações de maior contexto multiescala a partir de imagens de alta resolução espacial.

## 9.2 Limitações e trabalhos futuros

Apesar da rede proposta neste trabalho melhorar o desempenho quando há pequenas variações de GSD dentro de regiões do mosaico ou entre conjuntos de dados diferentes (capturados em alturas de voo variadas), todas as redes de segmentação multiescala ainda sofrem limitações quanto mais distante for o GSD treinado pela rede. Assim, outras estratégias podem ser utilizadas para incorporar atributos multiescala, como treinamento com imagens de entrada redimensionadas para várias escalas, que demandam um conjunto maior de dados rotulados.

No [Capítulo 8](#), foram realizados apenas análises qualitativas do desempenho, portanto ainda é necessário realizar mais experimentos com o modelo proposto, para analisar o desempenho da rede multiescala em um conjunto de imagens de treinamento com um GSD maior, ou seja, capturado em uma altura de voo mais elevada, com plantas daninhas pequenas no estágio inicial de crescimento. Um mapeamento aéreo com voos em alturas mais elevadas minimiza o número de imagens necessário para mapear uma grande área do terreno, com a vantagem de reduzir os requisitos computacionais para geração do mosaico, mas resulta em imagens de menor resolução com perda de detalhes dos objetos

de tamanhos menores. Com uma análise mais completa do modelo proposto, será possível determinar a necessidade de alterações nas taxas de dilatação dos filtros de convolução *atrous* do bloco denso para ajustar o campo de visão de acordo com a resolução das imagens de treinamento, lembrando que *kernels* de convolução com campo receptivo menor diminuem o contexto e maior destroem estruturas finas.

A obtenção de um modelo de segmentação semântica que possa incorporar diferentes estágios de crescimento das ervas daninhas continua sendo um desafio, especialmente devido ao fato de a aparência visual das plantas mudar significativamente durante o seu crescimento. Assim, para resolver esse problema em aplicações reais é necessário aumentar o conjunto de dados espaço-temporal de alta resolução, capturando imagens aéreas de outros campos agrícolas com plantas daninhas em vários estágios de crescimento. No entanto, dados representativos de ervas daninhas são mais difíceis de capturar do que da cultura devido à diversidade de espécies de daninhas que podem ser encontradas no campo.

Além disso, é necessário avaliar o desempenho usando outras CNNs como *backbone* da rede de segmentação multiescala proposta, tanto do ponto de vista de acurácia de predição quanto da complexidade computacional para obter predições de grandes mosaicos com um menor tempo de execução.

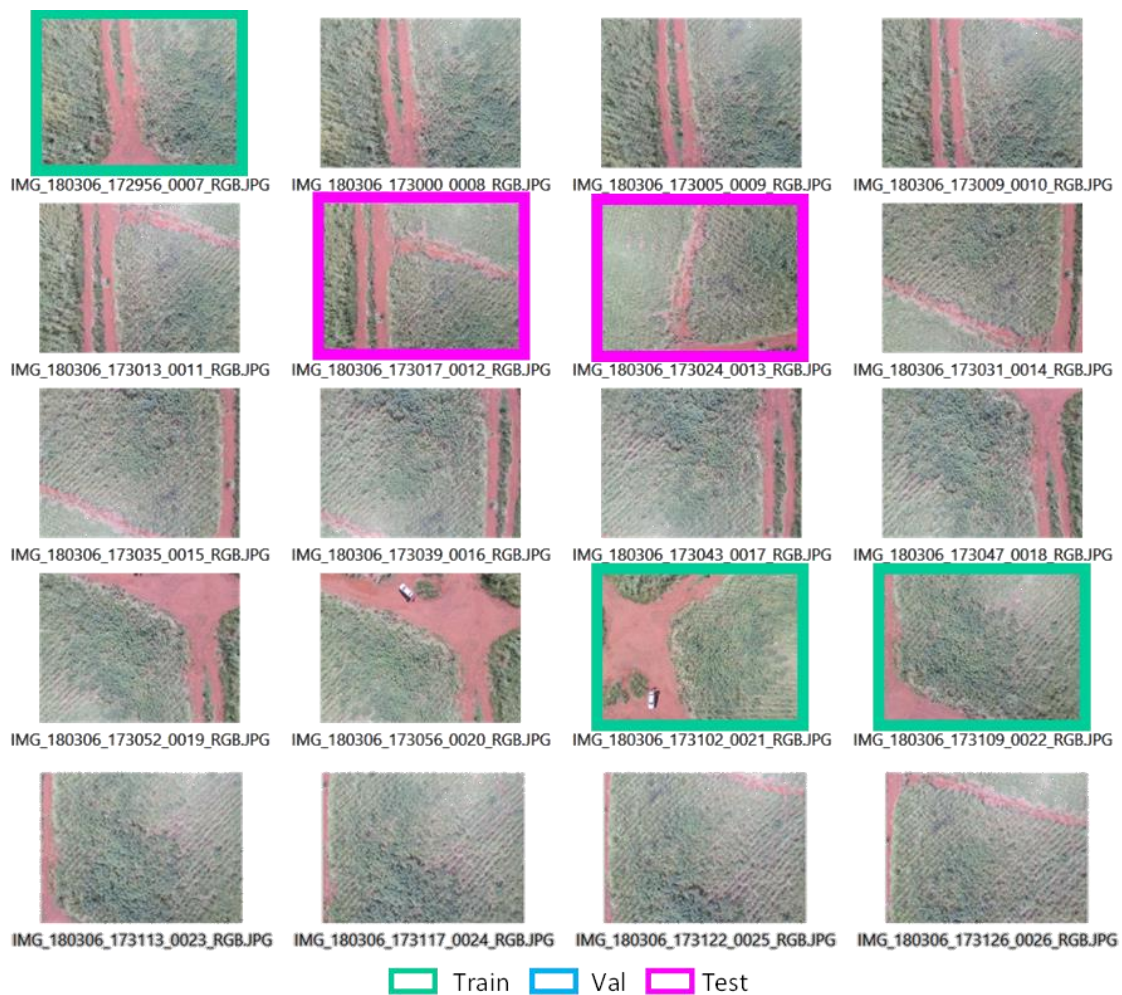
Por fim, como discutido no [Capítulo 2](#), a câmera multiespectral tem maior capacidade do que a câmera RGB para discriminar entre cultura e daninhas por meio da resposta do comportamento espectral dos alvos. Na [Seção 7.3](#), apesar do pequeno conjunto de dados de um mosaico multiespectral de baixa resolução, nossa proposta de rede multiescala obteve melhor resultado do que a rede BiSeNet (que não detectou plantas daninhas). Sendo assim, como sugestão para trabalhos futuros, uma análise mais detalhada das redes multiescala, usando mosaicos multicanais com diferentes combinações de bandas multiespectrais e mapas de índice de vegetação, pode melhorar o desempenho das redes de segmentação semântica em aplicações de agricultura.

Esperamos que este trabalho contribua com o desenvolvimento de novos métodos de segmentação semântica para gerar mapas de distribuição de plantas daninhas para agricultura de precisão. Essas informações serão importantes para criar mapas de prescrição, que podem ser transferidos para máquinas automatizadas, como pulverizadores de herbicidas. Desta forma, será possível minimizar o uso de produtos químicos e diminuir os impactos ambientais, mantendo a produtividade, para alcançar uma agricultura sustentável.

# Apêndice A

## Conjuntos de dados

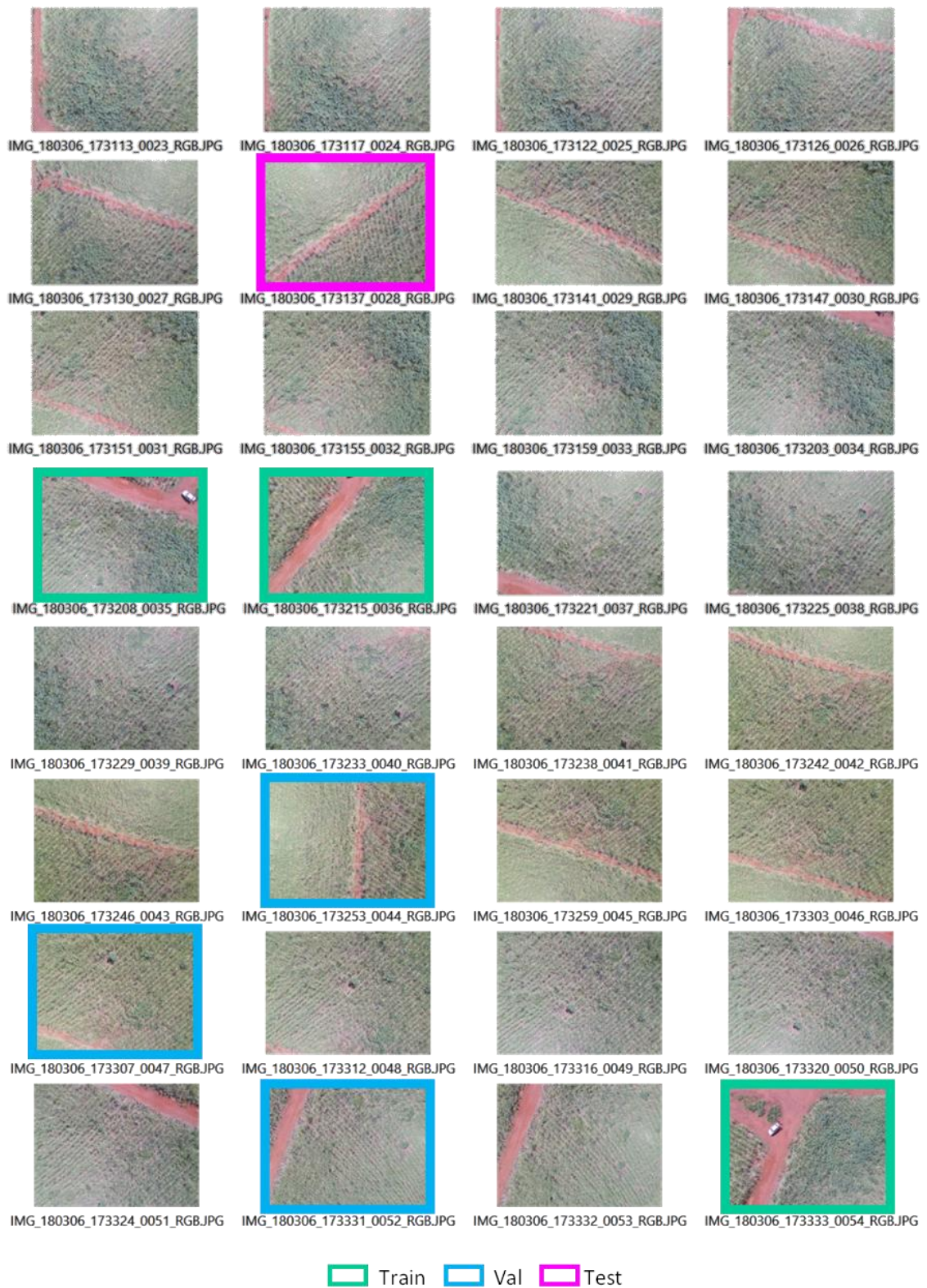
### A.1 Imagens RGB originais calibradas



**Figura A.1** – Seleção de 12 das 48 imagens RGB originais calibradas<sup>14</sup> de  $4.608 \times 3.456$  pixels para formar o conjunto de treinamento (vermelho), validação (azul) e teste (amarelo).

<sup>14</sup> Utilizamos apenas as imagens calibradas geometricamente, ou seja, as imagens capturadas que tiveram sua orientação interior e exterior (pose inicial) ajustadas durante o processo de geração do mosaico ortorretificado.





**Figura A.1** – Seleção de 12 das 48 imagens RGB originais calibradas de  $4.608 \times 3.456$  pixels para formar o conjunto de treinamento (vermelho), validação (azul) e teste (amarelo) (continuação).

## A.2 Imagens RGB selecionadas e *groundtruths*

Tabela A.1 – Imagens originais selecionadas para treinamento/validação e teste.

Imagem	Nome do arquivo JPEG ( <i>imagefile</i> )	Dataset
01	IMG_180306_172956_0007_RGB	Train
02	IMG_180306_173102_0021_RGB	
03	IMG_180306_173109_0022_RGB	
04	IMG_180306_173208_0035_RGB	
05	IMG_180306_173215_0036_RGB	
06	IMG_180306_173333_0054_RGB	
07	IMG_180306_173253_0044_RGB	Val
08	IMG_180306_173303_0047_RGB	
09	IMG_180306_173332_0052_RGB	
10	IMG_180306_173017_0012_RGB	Test
11	IMG_180306_173024_0013_RGB	
12	IMG_180306_173137_0028_RGB	

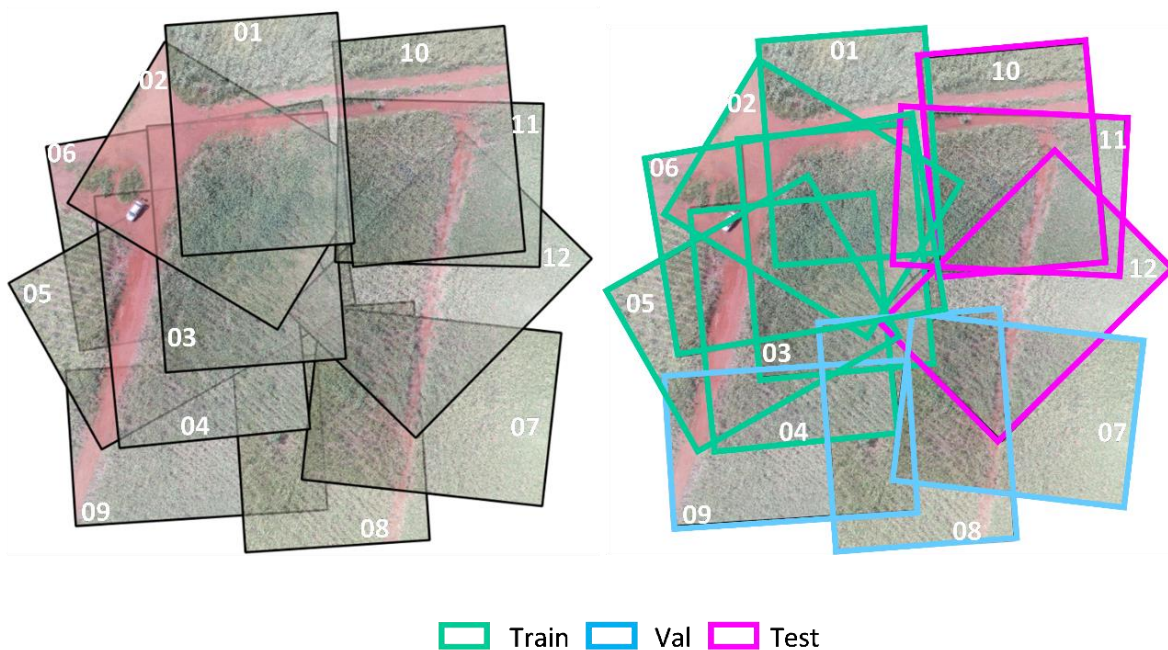
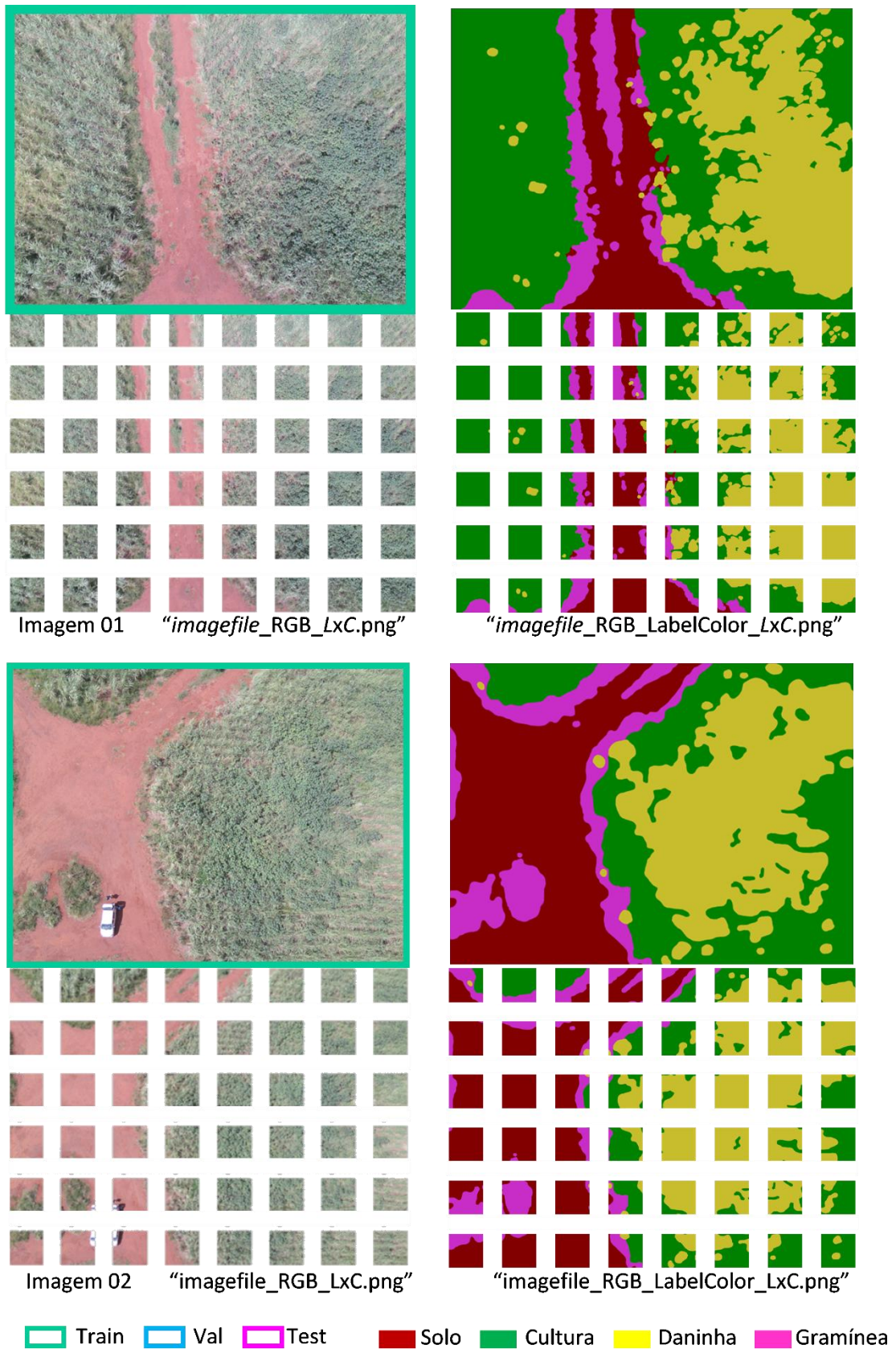
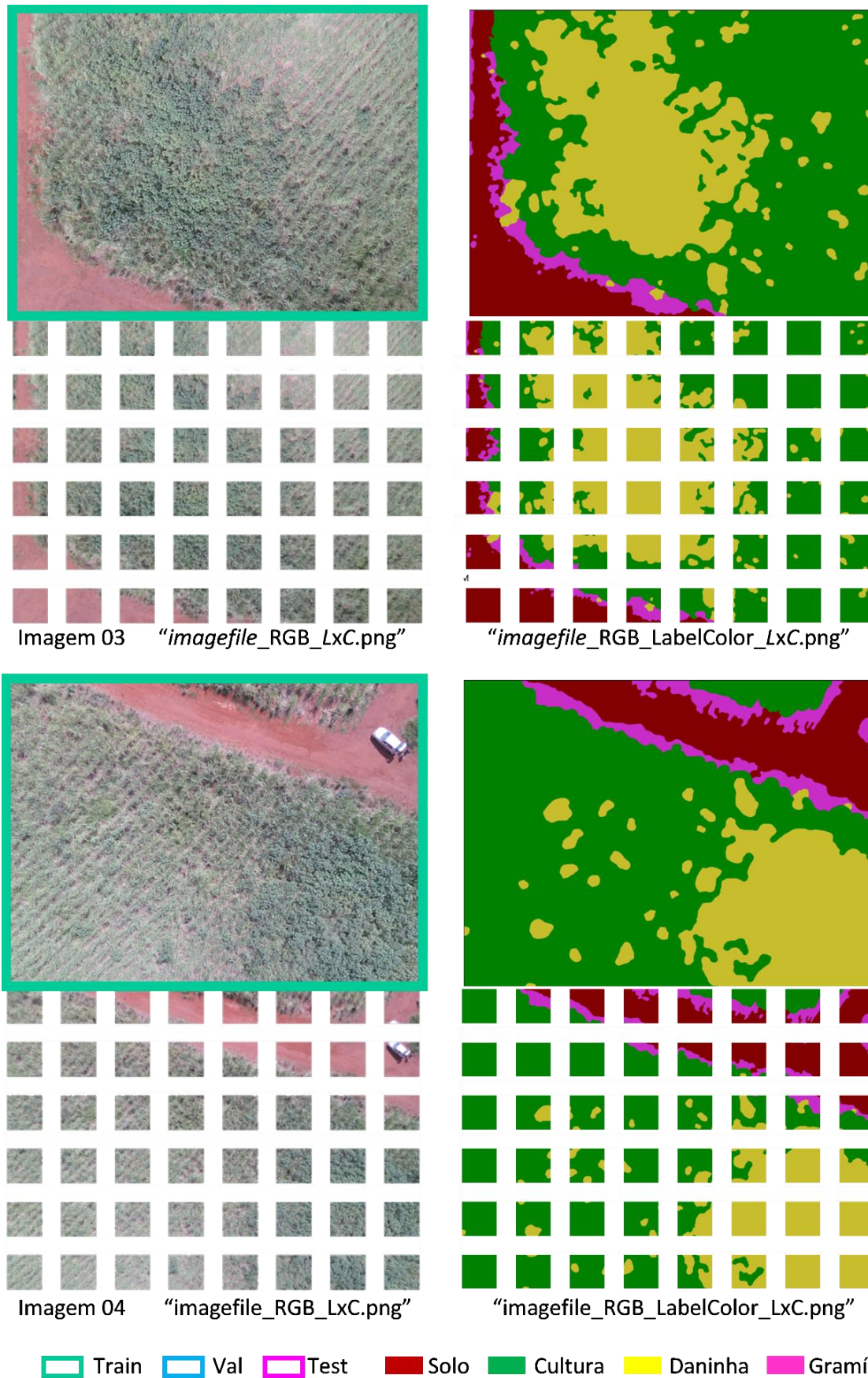


Figura A.2 – As 12 imagens RGB de  $4.608 \times 3.456$  pixels selecionadas para treinamento (vermelho), validação (azul) e teste (amarelo).



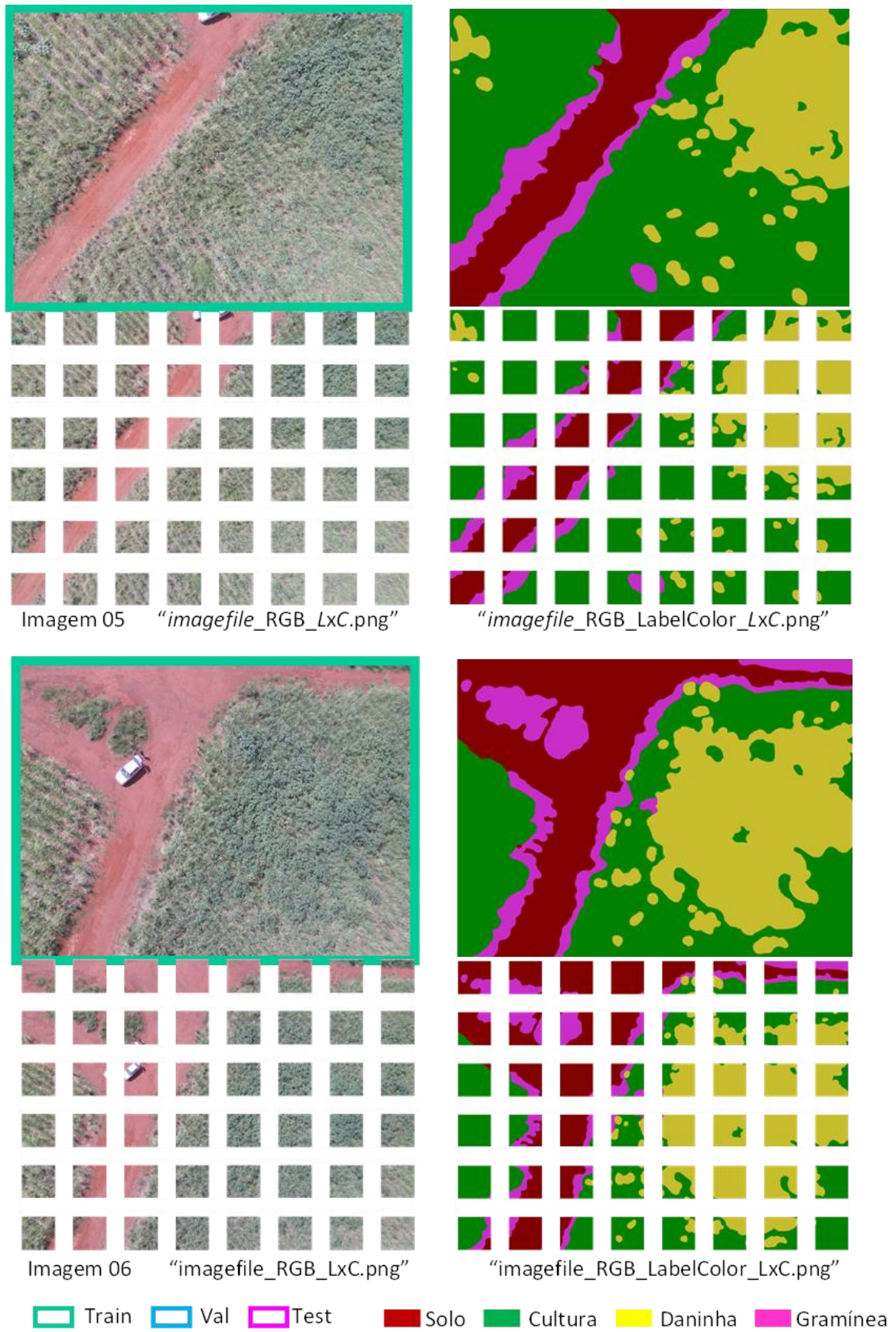


**Figura A.3** – *Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste.*

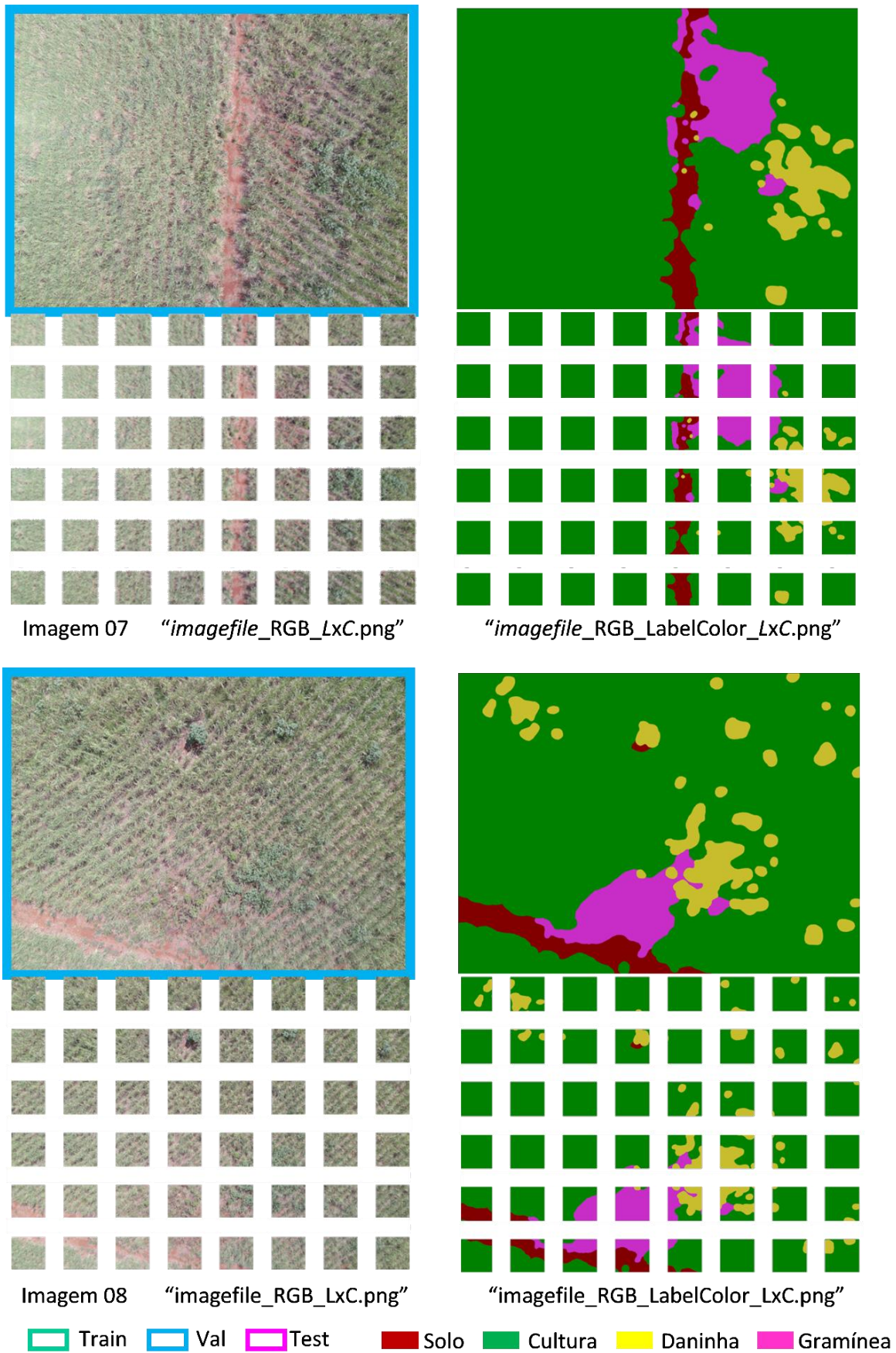


**Figura A.3** – Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste (continuação).



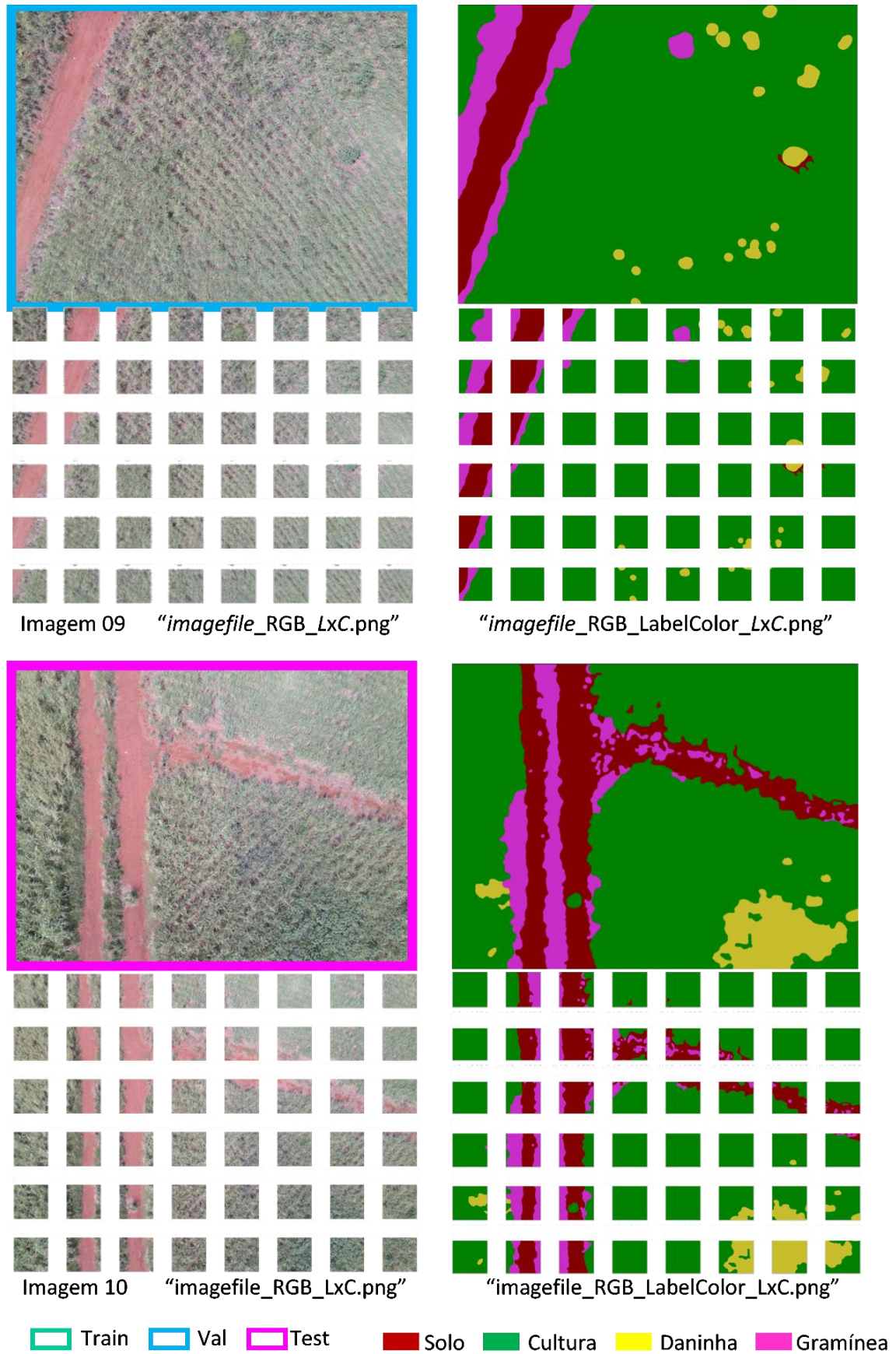


**Figura A.3** – Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste (continuação).



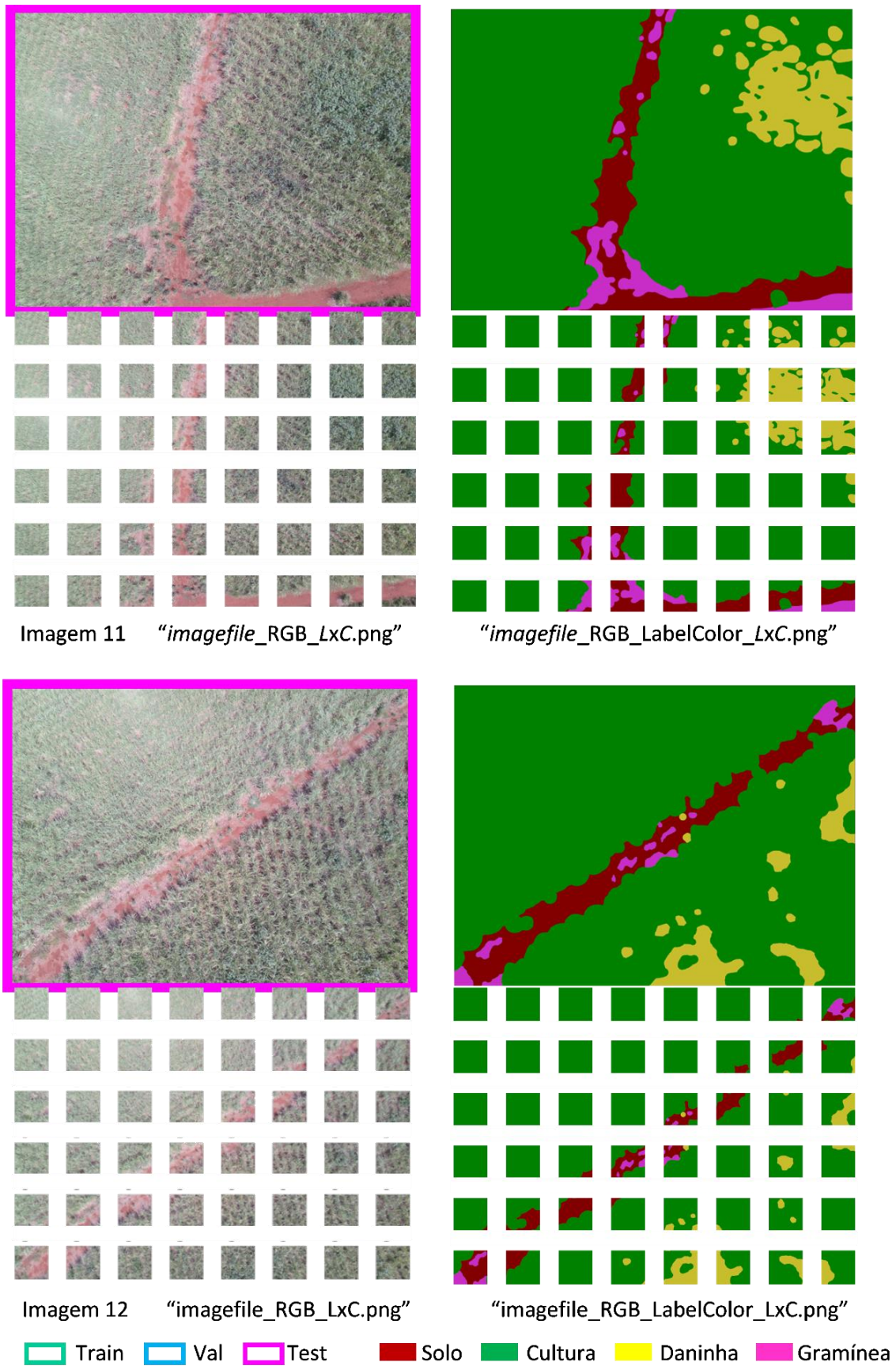
**Figura A.3** – Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste (continuação).





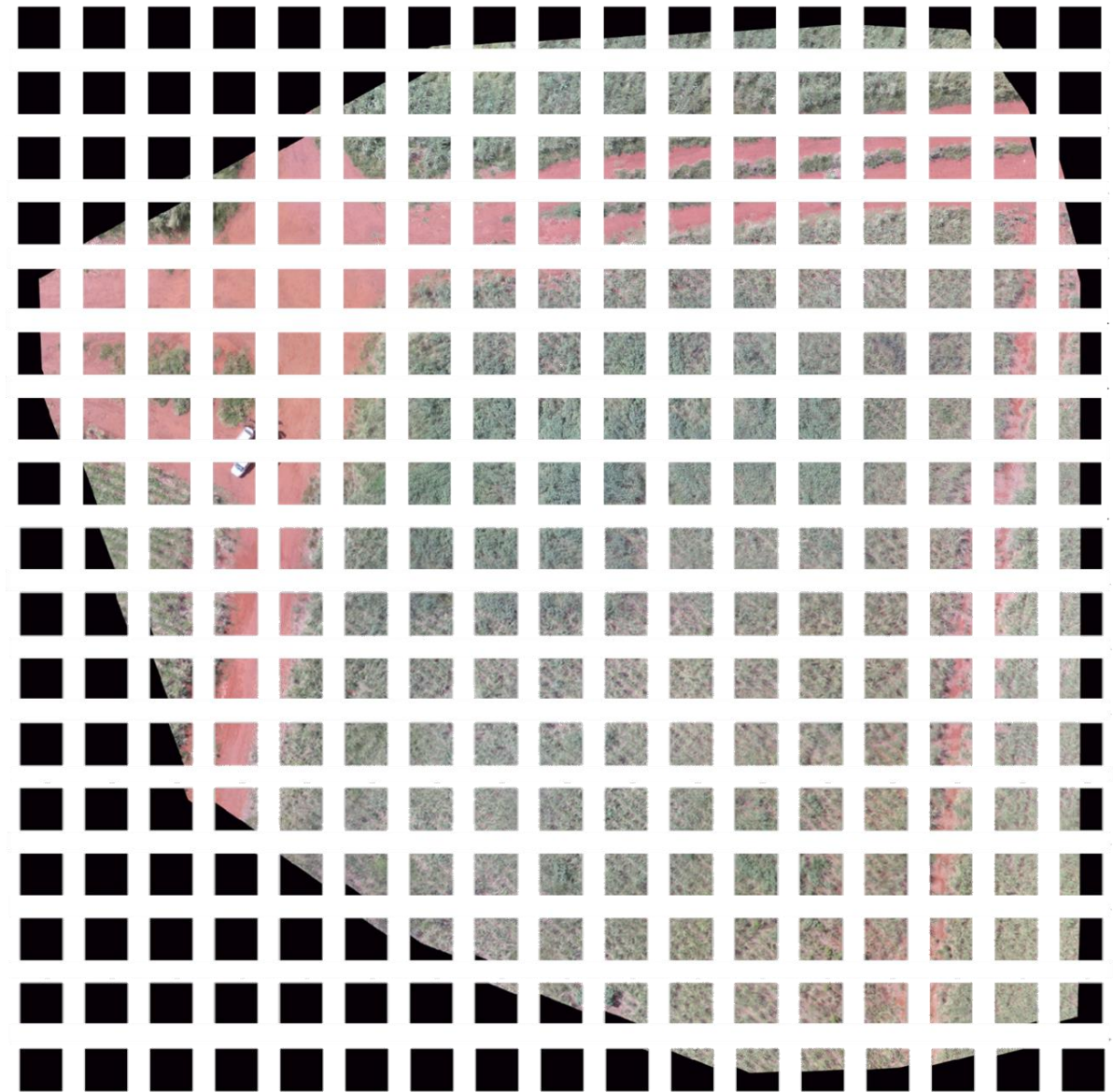
**Figura A.3** – *Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste (continuação).*





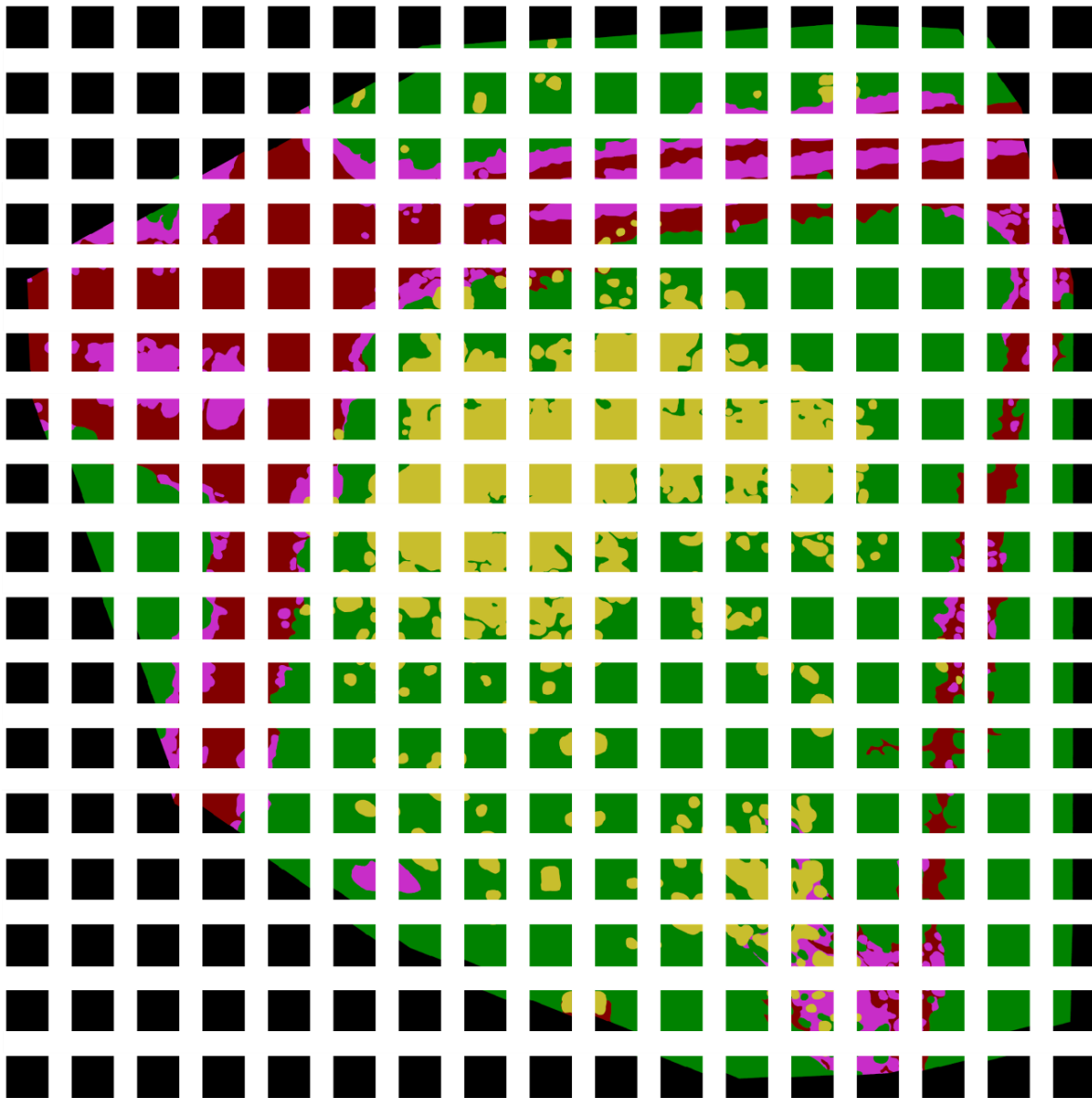
**Figura A.3** – Imagens originais RGB de  $4.608 \times 3.456$  pixels e groundtruths particionados em  $8 \times 6$  blocos de  $576 \times 576$  pixels para treinamento/validação/teste.

### A.3 Mosaico RGB e *groundtruth*



Blocos “mosaic\_RGB\_padding\_LxC.png”, com L e C de 1 a 17.

**Figura A.4** – Conjunto de teste com  $17 \times 17$  blocos de  $512 \times 512$  pixels do mosaico RGB, dos quais 100 blocos nas bordas apresentam fundo preto (valor de pixel zero).

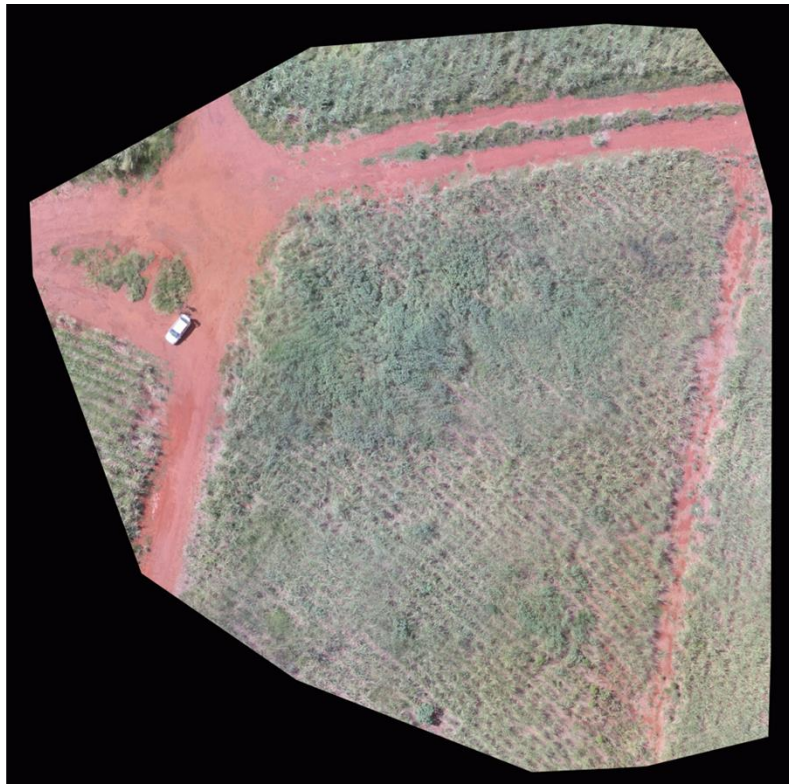


Blocos “mosaic\_RGB\_padding\_LabelColor\_LxC.png”, com L e C de 1 a 17.

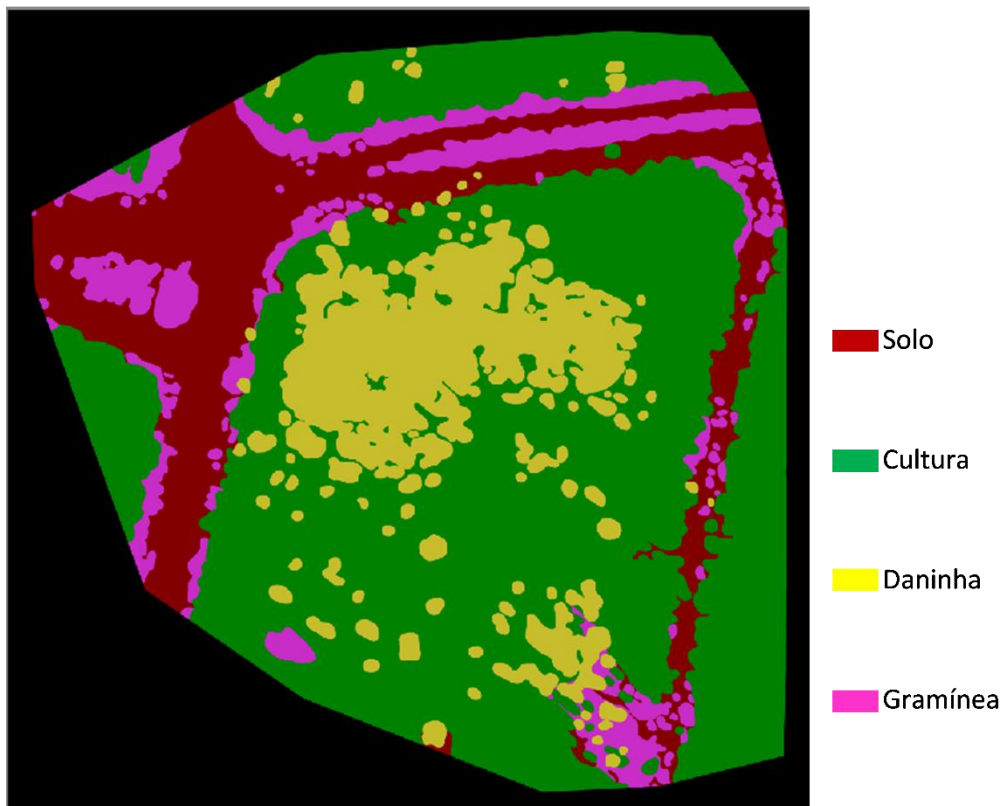
■ Solo   
 ■ Cultura   
 ■ Daninha   
 ■ Gramínea

**Figura A.5** – Conjunto de teste com  $17 \times 17$  blocos de  $512 \times 512$  pixels de groundtruths do mosaico RGB, dos quais 100 blocos nas bordas (de fundo preto) não são válidos para medida de desempenho.





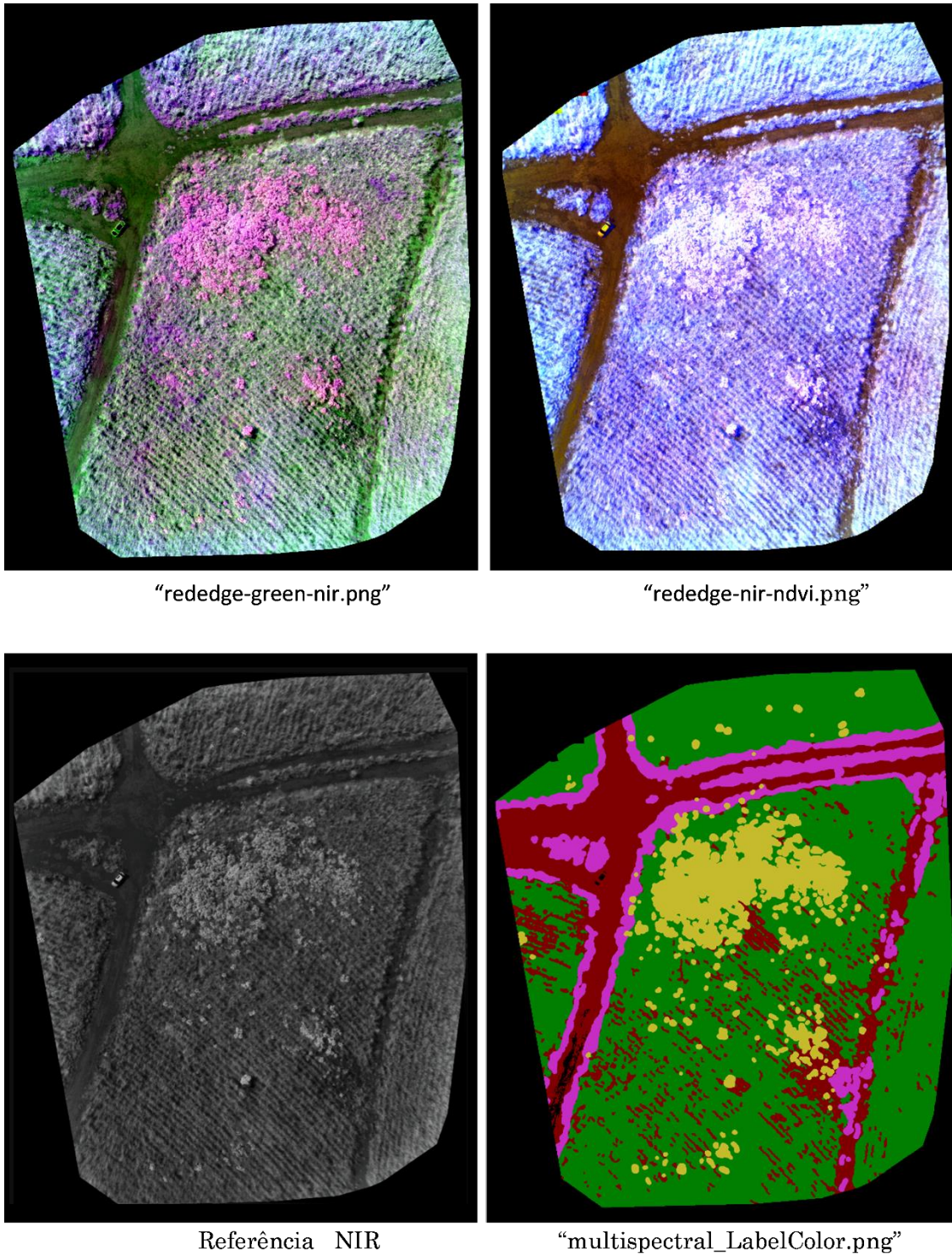
“mosaic\_RGB\_padding.png”



“mosaic\_RGB\_padding\_LabelColor.png”

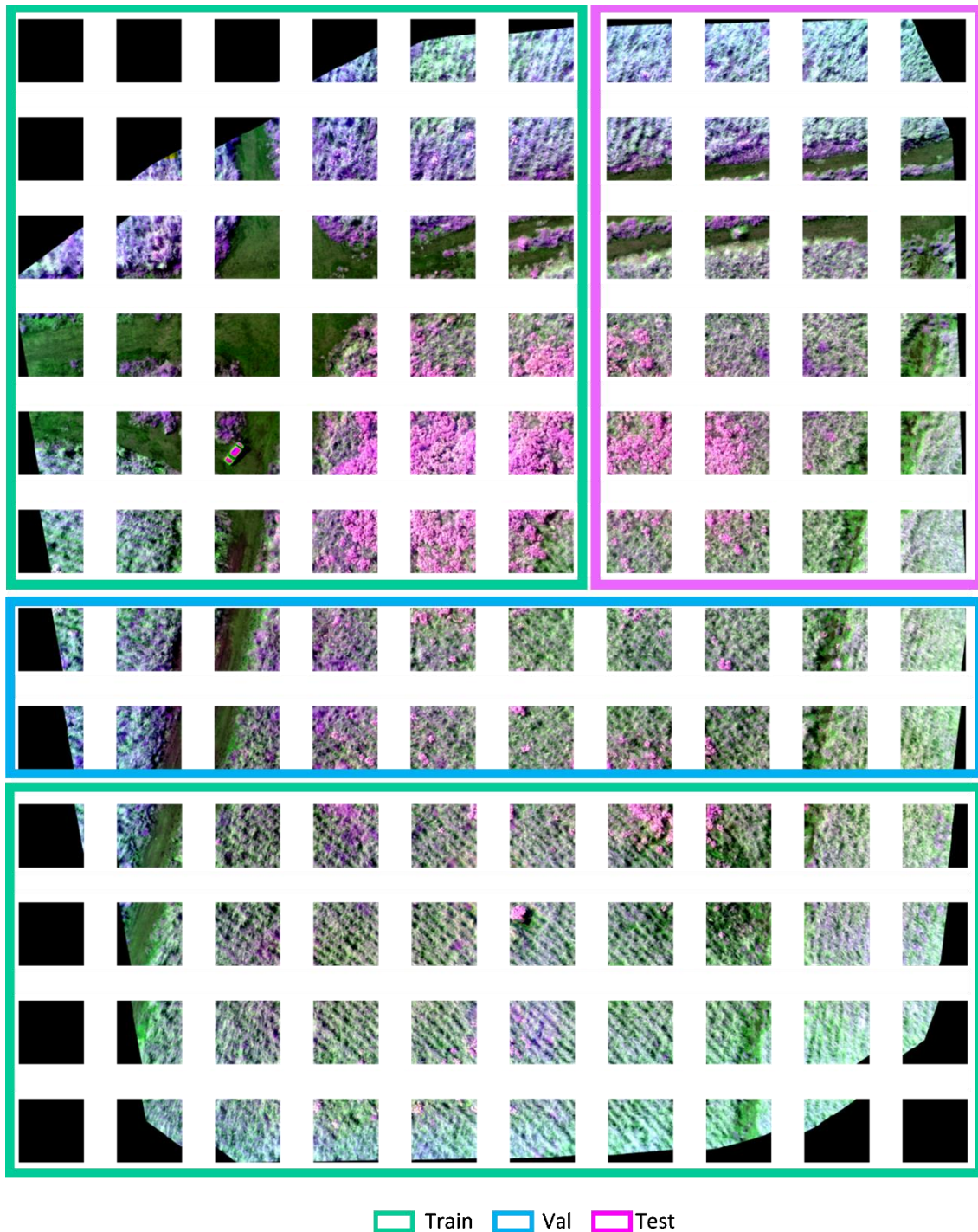
**Figura A.6** – Ortomosaico RGB expandido com preenchimento nas bordas e groundtruth, com  $8.704 \times 8.704$  pixels para teste das redes de segmentação semântica.

#### A.4 Composições multispectrais e *groundtruth*

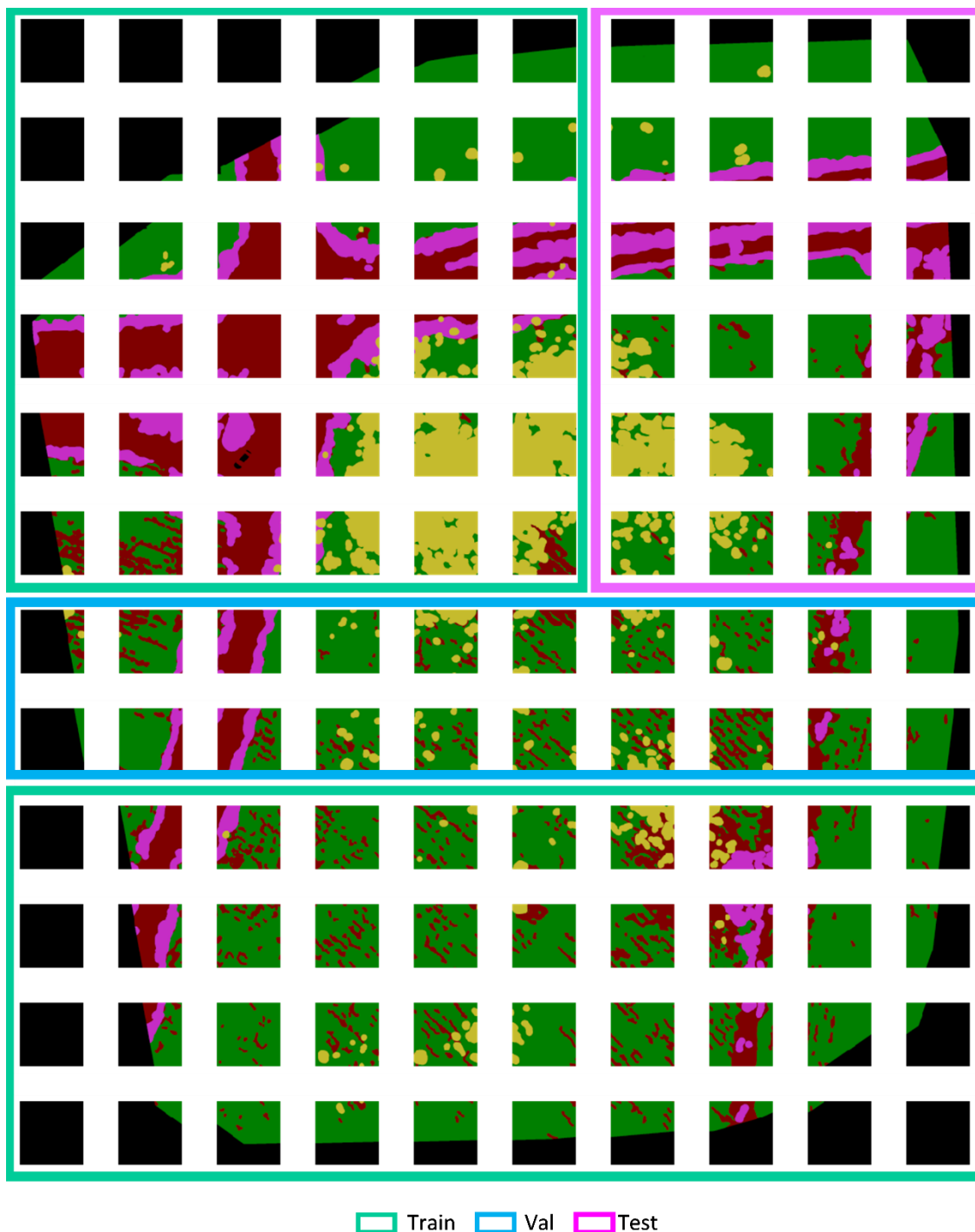


**Figura A.7** – Mapas multispectrais multicanais e *groundtruths* com preenchimento nas bordas, com  $2.560 \times 3.072$  pixels para treinamento e teste das redes de segmentação semântica. A banda NIR ou Red-Edge foi usada como referência para rotulação manual dos mapas multispectrais.





**Figura A.8** – Conjunto de dados com  $10 \times 12$  blocos de  $256 \times 256$  pixels do mapa RedEdge-Green-NIR, dos quais blocos inválidos nas bordas apresentam fundo preto (valor zero). Conjunto de treinamento com 44 blocos válidos (dentro das áreas retangulares verdes), validação com 16 blocos (azul), e teste com 15 blocos (lilás).



**Figura A.9** – Conjunto de dados com  $10 \times 12$  blocos de  $256 \times 256$  pixels de groundtruths do mapa RedEdge–Green–NIR, dos quais blocos com fundo preto são descartados para treinamento e medição de desempenho. Conjunto de treinamento com 44 blocos válidos (dentro das áreas retangulares verdes), validação com 16 blocos (azul), e teste com 15 blocos (lilás).



# Apêndice B

## Modelos de segmentação semântica

Nesta seção, são apresentados os modelos de segmentação semântica implementados no código de referência [LU 2019] utilizado neste trabalho. Os modelos de segmentação e os modelos-base disponíveis são:

MODEL = [FCN-8s, FCN-16s, FCN-32s, UNet, SegNet, Bayesian-SegNet, RefineNet, DeepLabV3, DeepLabV3Plus, DenseASPP, PSPNet, BiSegNet]

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169, DenseNet201, DenseNet264, MobileNetV1, MobileNetV2, Xception, Xception-DeepLab]

O programa de treinamento pode ser executado pelo comando

```
>> python train.py --model MODEL --base_model BASE_MODEL --dataset DATASET --num_classes NUM_CLASSES
```

com os seguintes argumentos:

--model	Modelo de segmentação semântica
--base_model	Modelo <i>backbone</i>
--dataset	Caminho do conjunto de dados
--loss {CE,Focal_Loss}	Função de perda para treinamento
--num_classes	Número de classes para segmentação
--random_crop	Recorte aleatório da imagem
--crop_height	Altura redimensionada da imagem
--crop_width	Largura redimensionada da imagem
--batch_size	Tamanho do lote de treinamento
--valid_batch_size	Tamanho do lote de validação
--num_epochs	Número de épocas para treinamento
--initial_epoch	Época inicial de treinamento
--h_flip	Espelhamento horizontal aleatório da imagem
--v_flip	Espelhamento vertical aleatório da imagem
--brightness	Mudança de brilho aleatório da imagem (lista)
--rotation	Ângulo de rotação aleatório da imagem
--zoom_range	Faixa de zoom
--channel_shift	Deslocamento de canal
--data_aug_rate	Taxa de aumento artificial de dados
--checkpoint_freq	Frequência para salvar um checkpoint
--validation_freq	Frequência para executar a validação
--num_valid_images	Número de imagens usadas para validação

<code>--data_shuffle</code>	Embaralhamento dos dados
<code>--random_seed</code>	
<code>--weights</code>	Caminho dos pesos a carregar
<code>--optimizers</code>	Otimizadores
<code>--learning_rate</code>	Taxa de aprendizado
<code>--lr_scheduler</code>	Estratégia de taxa de aprendizado

O programa de teste pode ser executado pelo comando:

```
>> python test.py --model MODEL --base_model BASE_MODEL --dataset DATASET --num_classes
NUM_CLASSES --weights "weights_path"
```

O programa para predição das imagens pode ser executado pelo comando:

```
>> python predict.py --model MODEL --base_model BASE_MODEL --num_classes NUM_CLASSES --
weights "weights_path" --image_path "image_path"
```

O programa para avaliação de todas as classes definidas no arquivo “*evaluated\_classes.txt*” a partir das predições das imagens de teste pode ser executado pelo comando:

```
>> python evaluate.py --dataset DATASET --predictions 'prediction_path'
```

O programa suporta as funções de perda (*loss*):

- Cross entropy;
- Focal loss;
- MIoU loss;
- Self balanced focal loss,

os otimizadores (*optimizers*):

- SGD;
- Adam;
- Nadam;
- AdamW;
- NadamW;
- SGDW,

e as estratégias para taxa de aprendizado (*lr\_scheduler*):

- *step decay*;
- *poly decay*;
- *cosine decay*;
- *warm up*.

A métrica de desempenho da rede é o mIoU, descrito no [Apêndice C](#).

A seguir são apresentados os modelos das redes de segmentação semântica baseados em CNNs, implementados de acordo com as informações apresentadas nas [Seções 2.3 e 2.4](#) do [Capítulo 2](#). Cada modelo de rede suporta todos ou alguns *backbones* específicos, com pequenas alterações nas saídas utilizadas, dependendo do *backbone* selecionado.



## B.1 Modelo FCN

MODEL = {FCN-8s, FCN-16s, FCN-32s}

BASE\_MODEL = VGG16 (default)

Para o modelo FCN-32s, é utilizada a saída c5 de todos os *backbones* disponíveis.

$c5 = backbone (inputs, output = ['c5'])$

Para o modelo FCN-16s, se BASE\_MODEL for: (a) VGG16, VGG19, ResNet50, ResNet101, ResNet152, MobileNetV1 ou MobileNetV2 são utilizadas as saídas c4 e c5 do *backbone*; (b) DenseNet121, DenseNet169, DenseNet201, DenseNet264, Xception, Xception-DeepLab, são utilizadas as saídas c3 e c5 do *backbone*.

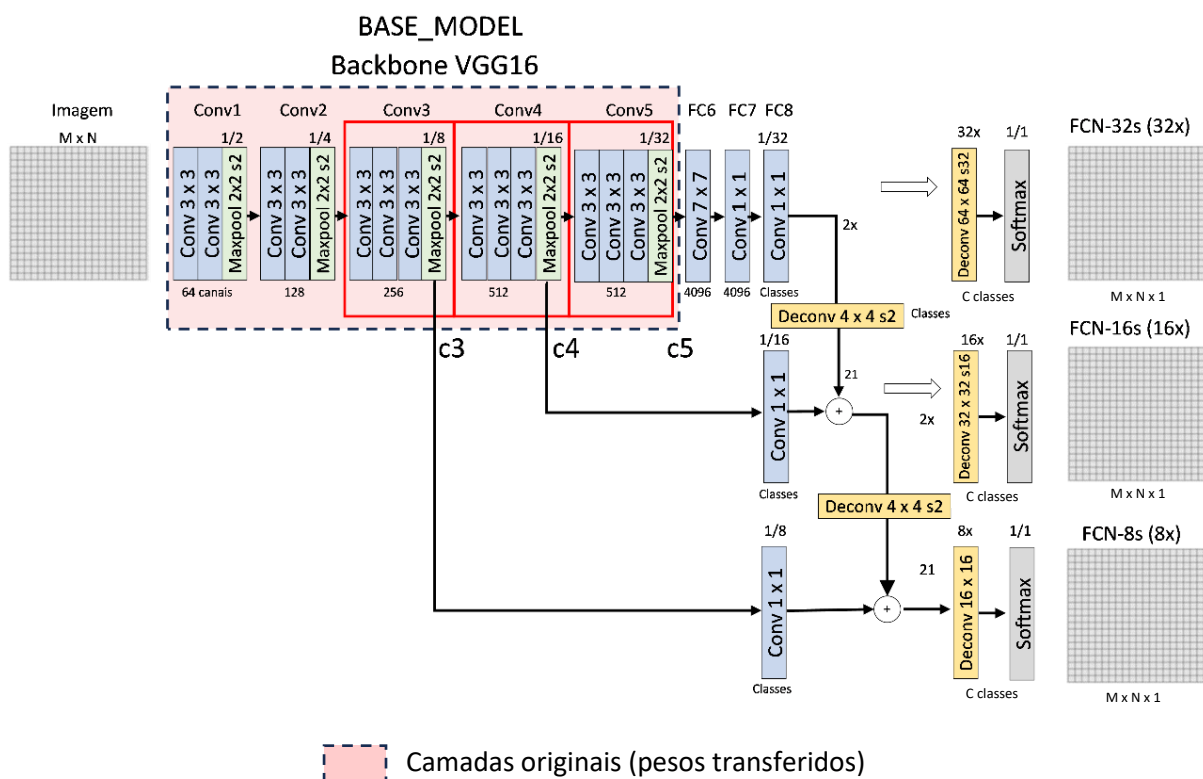
(a)  $c4, c5 = backbone (inputs, output = ['c4', 'c5'])$

(b)  $c4, c5 = backbone (inputs, output = ['c3', 'c5'])$

Para o modelo FCN\_8s, se BASE\_MODEL for: (a) VGG16, VGG19, ResNet50, ResNet101, ResNet152, MobileNetV1, MobileNetV2, são utilizadas as saídas c3, c4, c5 do *backbone*; (b) DenseNet121, DenseNet169, DenseNet201, DenseNet264, Xception ou Xception-DeepLab, são utilizadas as saídas c2, c3, c5.

(a)  $c3, c4, c5 = backbone (inputs, output = ['c3', 'c4', 'c5'])$

(b)  $c3, c4, c5 = backbone (inputs, output = ['c2', 'c3', 'c5'])$



**Figura B.1** – Modelo FCN-32s, 16s e 8s com backbone VGG-16.

## B.2 Modelo UNet

MODEL = UNet

BASE\_MODEL = VGG16 (default)

Para o modelo UNet, os *backbones* permitidos são:

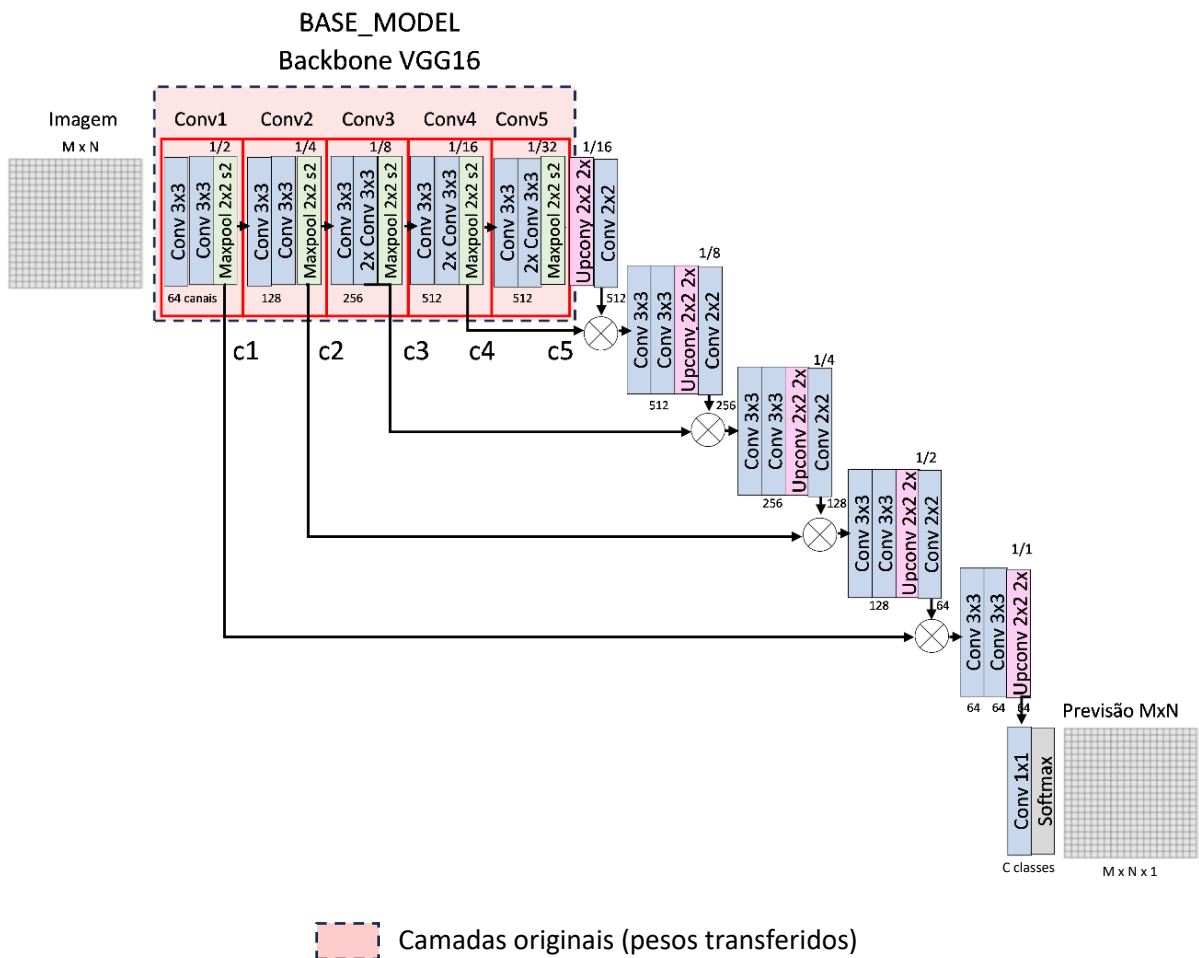
BASE\_MODEL = [VGG16, VGG19, MobileNetV1, MobileNetV2].

Todas as saídas do *backbone* são utilizadas:

$c1, c2, c3, c4, c5 = \text{backbone}(\text{inputs}, \text{output} = ['c1', 'c2', 'c3', 'c4', 'c5'])$

A primeira versão da rede U-Net original, mostrada na [Figura 2.71](#), utiliza a rede VGG-13 como *backbone*. Além disso, as saídas do *backbone* para conexões de salto não são subamostradas na camada *maxpooling*.

No código de referência, o caminho de expansão da rede U-Net tem as operações de *upsampling* seguidas de uma convolução  $2 \times 2$ , concatenação com a saída correspondente do *backbone*, e duas convoluções  $3 \times 3$  seguidas de ReLU. As conexões de salto são subamostradas.



**Figura B.2** – Modelo U-Net com backbone VGG-16.

## B.3 Modelo SegNet

MODEL = [SegNet, Bayesian-SegNet]

BASE\_MODEL = VGG16 (default)

Para o modelo SegNet, todos os *backbones* podem ser utilizados:

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169, DenseNet201, DenseNet269, MobileNetV1, MobileNetV2, Xception, Xception-DeepLab]

A saída *c5* do *backbone* é utilizada pelo decodificador da rede SegNet. Não foi implementado com as conexões de índices de *pooling*.

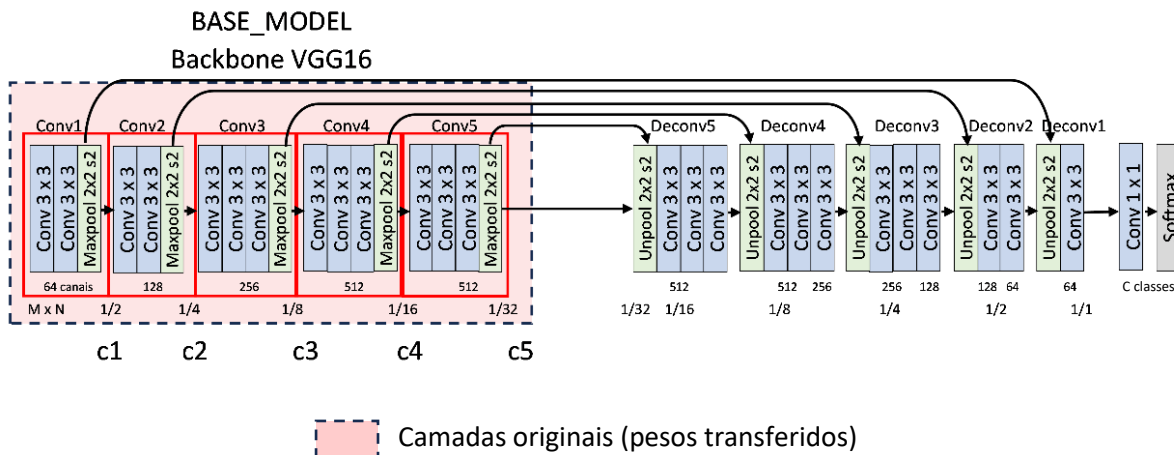


Figura B.3 – Modelo SegNet com backbone VGG-16.

## B.4 Modelo RefineNet

MODEL = RefineNet

BASE\_MODEL = ResNet50

No modelo RefineNet, os *backbones* disponíveis são:

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, MobileNetV1, MobileNetV2].

As saídas c2, c3, c4 e c5 são utilizadas pela RefineNet.

c1, c2, c3, c4, c5 = *backbone* (*inputs*, *output* = ['c2', 'c3', 'c4', 'c5'])

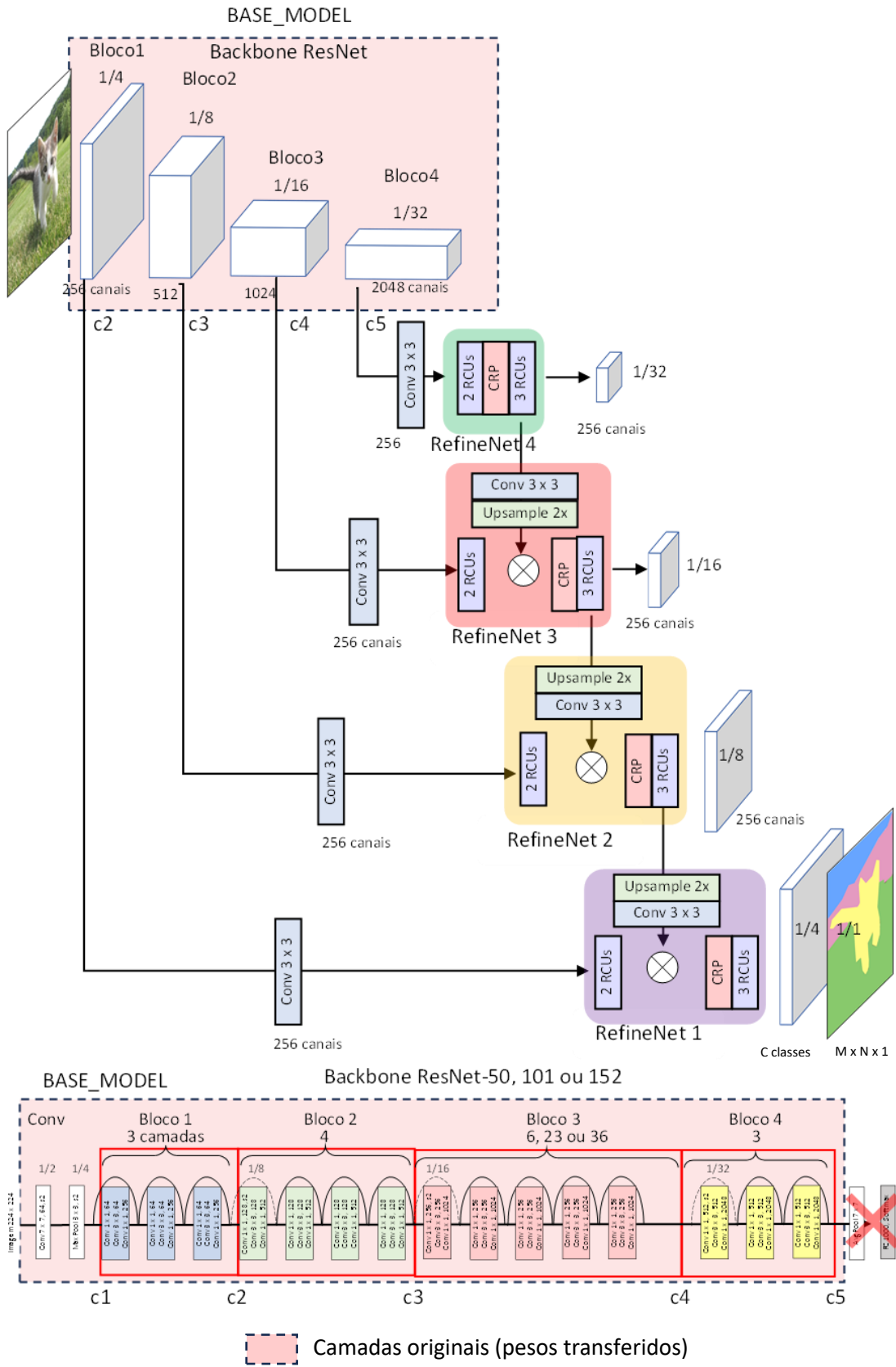


Figura B.4 – Modelo RefineNet com backbone ResNet-50.

## B.5 Modelo BiSegNet

MODEL = BiSegNet

BASE\_MODEL = Xception

No modelo BiSegNet, todos os *backbones* podem ser utilizados:

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, MobileNetV1, MobileNetV2, Xception, Xception-DeepLab]

No caminho de contexto, se o *backbone* for: (a) VGG16, VGG19, ResNet50, ResNet101, ResNet152, MobileNetV1, MobileNetV2, Xception, Xception-DeepLab, as saídas c4 e c5 do *backbone* são as entradas dos módulos ARM (*Attention Refinement Module*); (b) se for DenseNet121, DenseNet169, DenseNet201, DenseNet264, as saídas c3 e c5 são usadas.

c4, c5 = *backbone* (inputs, outputs = ['c4', 'c5'])

c4, c5 = *backbone* (inputs, outputs = ['c3', 'c5'])

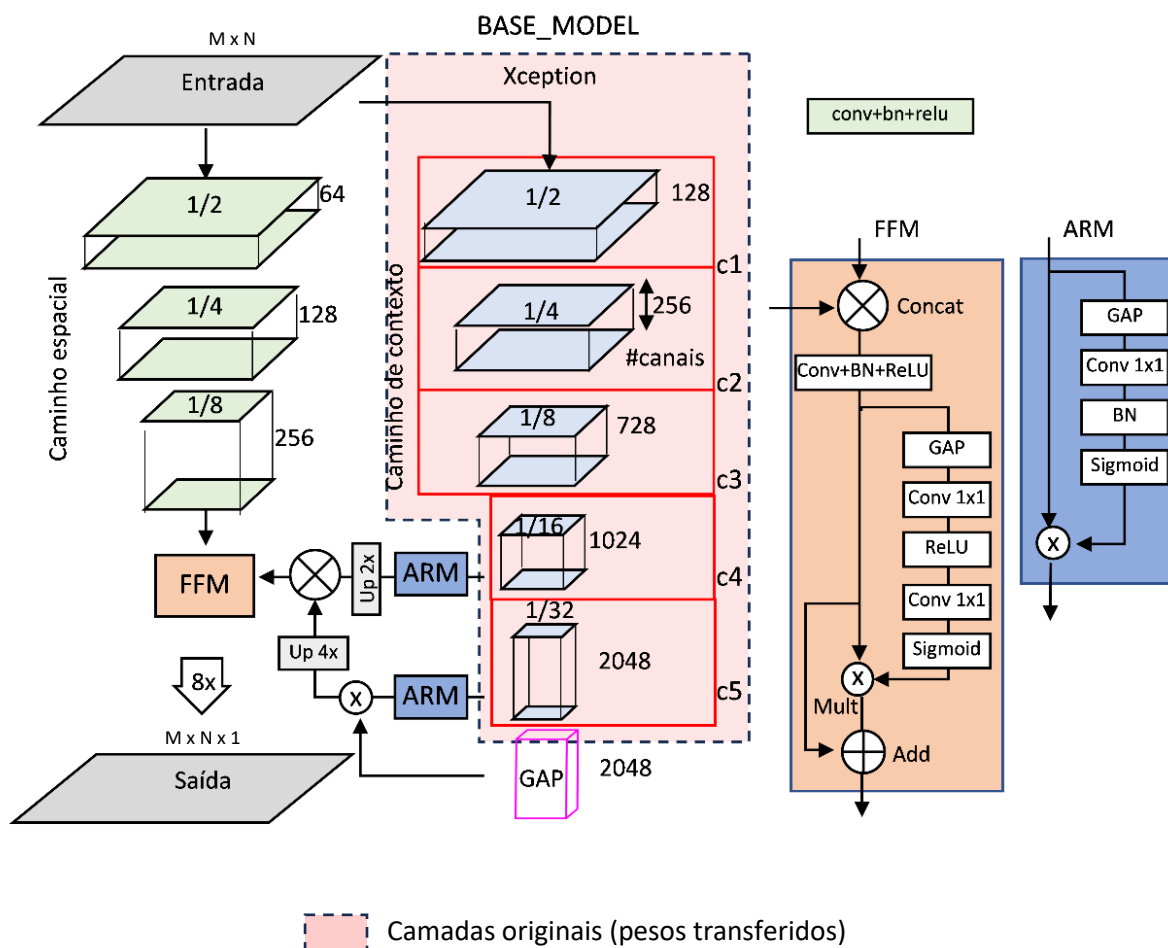


Figura B.5 – Modelo BiSegNet com backbone Xception.



## B.6 Modelo PSPNet

MODEL = PSPNet

BASE\_MODEL = ResNet50 (default)

No modelo PSPNet, todos os *backbones* podem ser utilizados. A saída *c5* do *backbone* é usada na entrada da rede PSPNet.

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169, DenseNet201, DenseNet264, MobileNetV1, MobileNetV2, Xception-DeepLab]

A rede *backbone* usa convolução *atrous* com dilatação [2, 4] nos dois últimos blocos convolucionais, removendo a subamostragem para obter uma saída de 1/8 do tamanho da entrada.

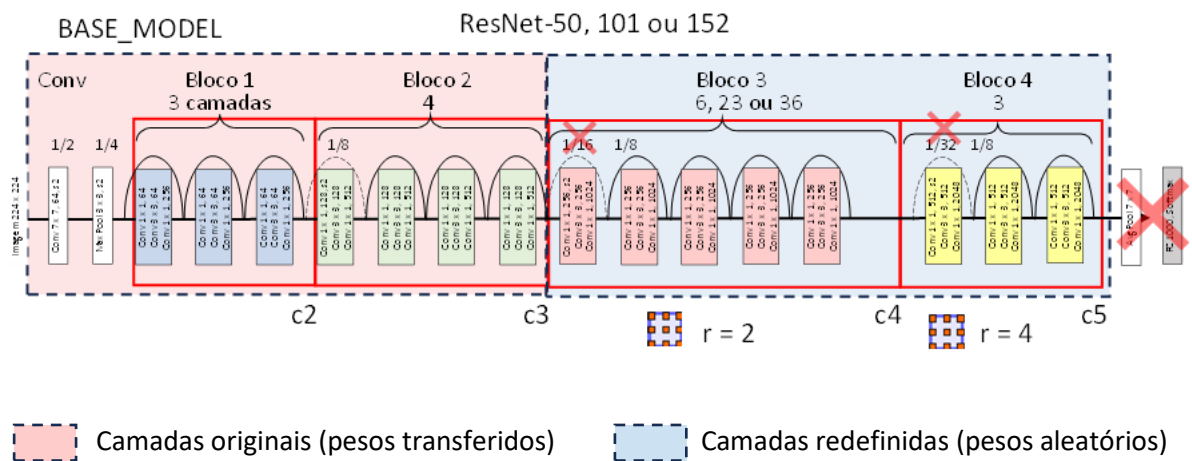
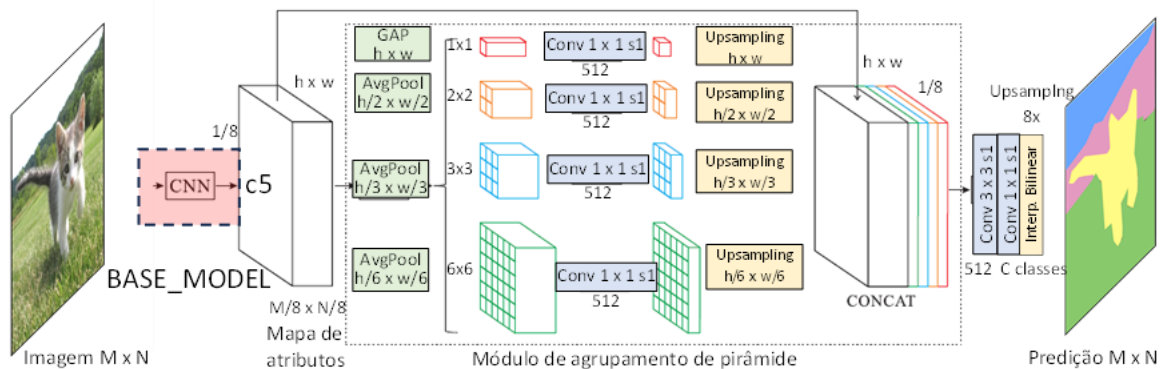


Figura B.6 – Modelo PSPNet com backbone ResNet-50.

## B.7 Modelo DeepLabV3

MODEL = DeepLabV3

BASE\_MODEL = ResNet50 (default)

No modelo DeepLabV3, todos os *backbones* podem ser utilizados, exceto Xception (que não implementa convolução *atrous*), substituído por Xception-DeepLab:

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169, DenseNet201, DenseNet264, MobileNetV1, MobileNetV2, Xception-DeepLab]

A saída *c4* do *backbone* é utilizada pela rede DeepLabV3, que redefine o Bloco 4 do *backbone* para três camadas convolucionais residuais de mapeamento e identidade com taxas de dilatação [1, 2, 4].

$x = backbone (inputs, output='c4')$

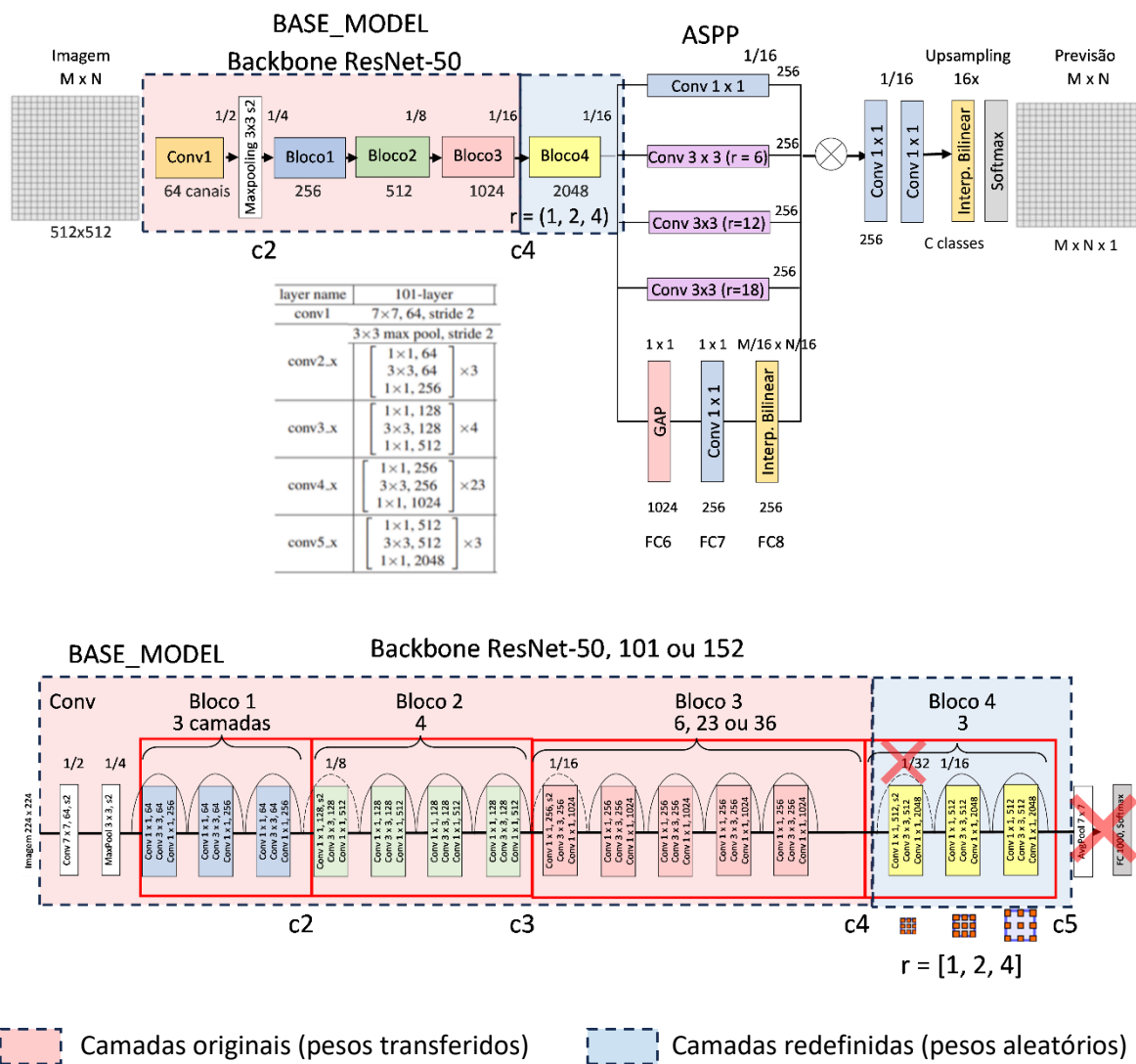


Figura B.7 – Modelo DeepLab-v3 com backbone ResNet-50.





## B.9 Modelo DenseASPP

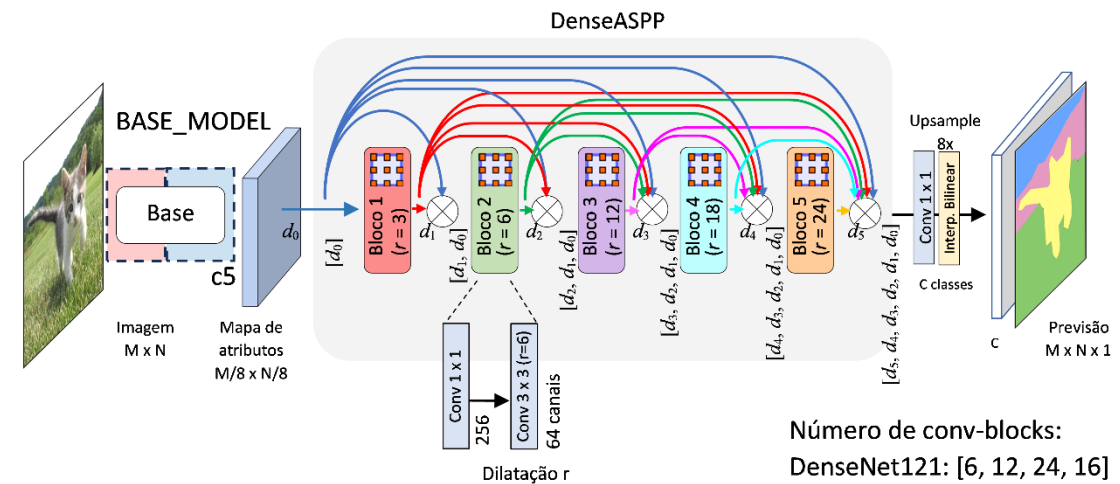
MODEL = DenseASPP

BASE\_MODEL = DenseNet121 (default)

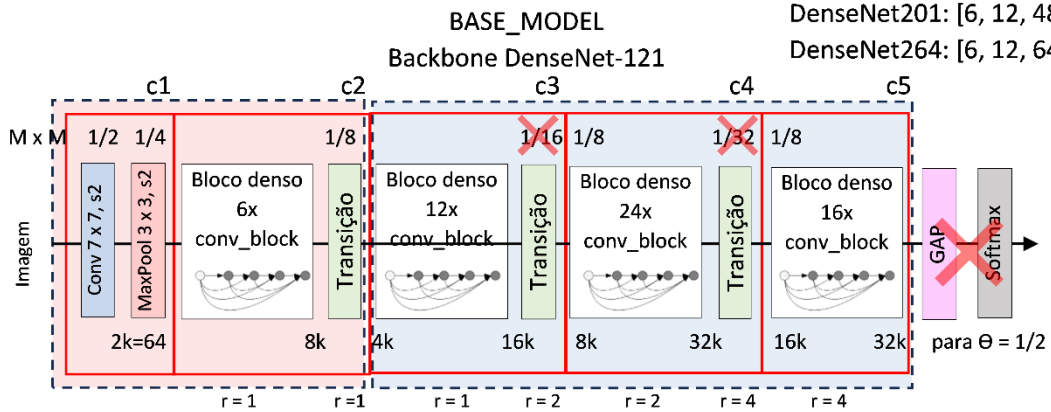
No modelo DenseASPP, todos os *backbones* podem ser utilizados:

BASE\_MODEL = [VGG16, VGG19, ResNet50, ResNet101, ResNet152, DenseNet121, DenseNet169, DenseNet201, DenseNet264, MobileNetV1, MobileNetV2, Xception-DeepLab]

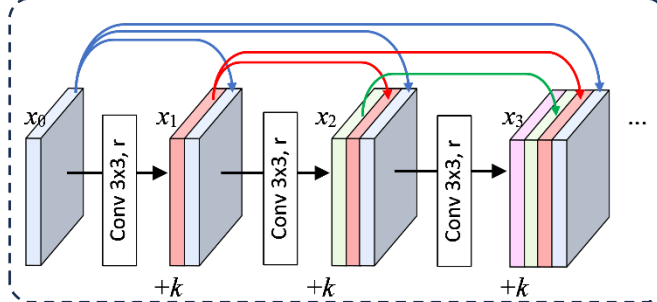
A saída *c5* do *backbone* é usado como entrada da rede DenseASPP.



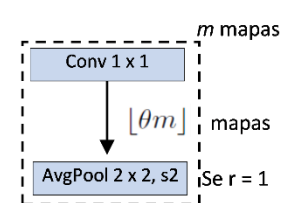
Número de conv-blocks:  
 DenseNet121: [6, 12, 24, 16]  
 DenseNet169: [6, 12, 32, 32]  
 DenseNet201: [6, 12, 48, 32]  
 DenseNet264: [6, 12, 64, 48]



Bloco denso com  $n$  conv-blocks com taxa de crescimento  $k = 32$



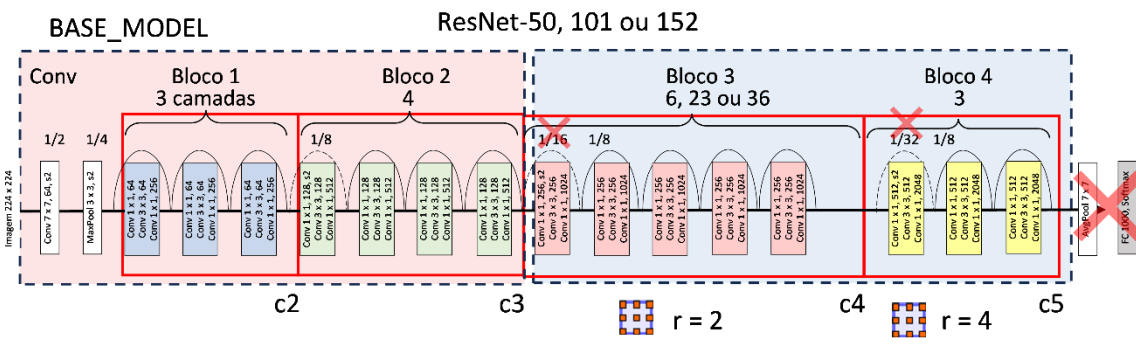
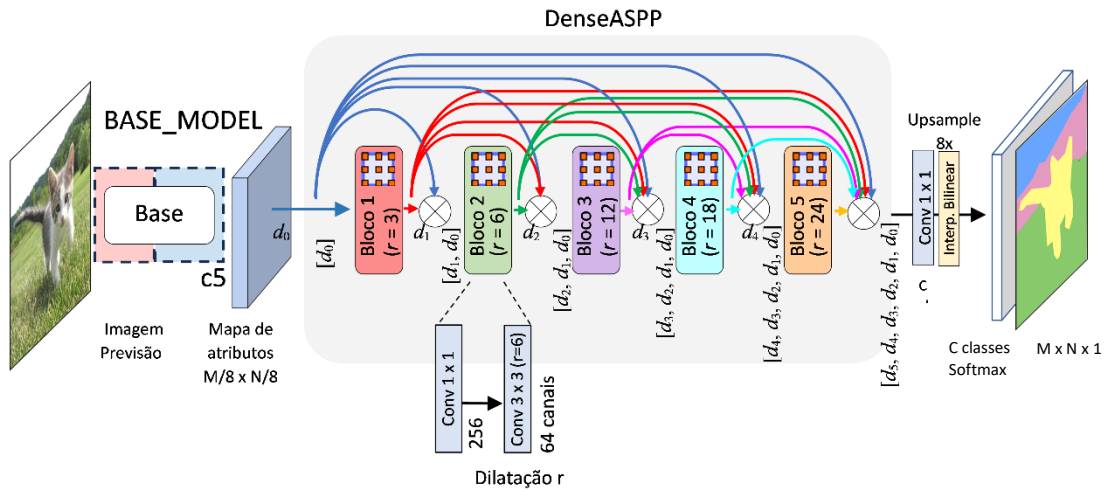
Camada de transição



  Camadas originais (pesos transferidos)        Camadas redefinidas (pesos aleatórios)

Figura B.11 – Modelo DenseASPP com backbone DenseNet-121.





  Camadas originais (pesos transferidos)    
   Camadas redefinidas (pesos aleatórios)

**Figura B.12** – Modelo DenseASPP com backbone ResNet-50.



# Apêndice C

## Métricas de desempenho

Métricas são ferramentas estatísticas usadas para comparar e avaliar os classificadores. Para tarefas de segmentação semântica densa, existem muitas métricas de avaliação de desempenho, como *Pixel Accuracy* (PA), *Mean Pixel Accuracy* (MPA), *Mean Intersection over Union* (mIoU) e *Frequency Weighted Intersection over Union* (FWIoU). Neste projeto, utilizamos a métrica mIoU entre todas as classes para avaliar o desempenho dos modelos de rede utilizados.

### C.1 Matriz de confusão

Em aprendizado de máquina, uma matriz de confusão ou matriz de erro é uma tabela que permite a visualização do desempenho de um algoritmo de classificação de aprendizado supervisionado. A tabela tem duas dimensões (classe real e classe prevista) e conjuntos idênticos de classes em ambas as dimensões. Assim, cada linha da matriz representa as instâncias de uma classe real, enquanto cada coluna representa as instâncias de uma classe predita (ou vice-versa). Uma célula na linha  $i$  e na coluna  $j$  em uma matriz de confusão contém o número de amostras no conjunto de dados que pertencem à classe  $C_i$  e foram classificadas como classe  $C_j$ . A matriz de confusão permite perceber se o classificador está confundindo duas classes (ou seja, rotulando uma como outra).

No caso de classificação binária, com uma amostra sendo classificada como pertencente a uma classe ou não, a matriz pode ser representada na [Figura C.1](#), onde:

- Verdadeiro positivo (*true positive* — TP): ocorre quando uma instância da classe real que estamos buscando prever foi prevista corretamente.
- Falso positivo (*false positive* — FP): ocorre quando uma instância que não pertence a classe real que estamos buscando prever foi prevista incorretamente.
- Verdadeiro negativo (*true negative* — TN): ocorre quando uma instância que não pertence a classe real que estamos buscando prever foi prevista corretamente.
- Falso negativo (*false negative* — FN): ocorre quando uma instância que pertence a classe que estamos buscando prever foi prevista incorretamente.

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Figura C.1 – Matriz de confusão.

Através da matriz de confusão são calculadas outras derivações [MICROSOFT]:

- 1) Condição positiva (P): número de casos positivos reais (TP + FN);
- 2) Condição negativa (N): número de casos negativos reais (FP + TN);
- 3) Recall (sensibilidade, taxa de acerto ou taxa positiva verdadeira (TPR):

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

Mede a capacidade de um modelo classificar todas as amostras positivas, ou seja, qual o percentual de casos positivos reais (P) que são previstos corretamente como positivos. A métrica *recall* é definida como a razão entre verdadeiros positivos (TP) sobre a soma de verdadeiros positivos (TP) com falsos negativos (FN).

Objetivo: Quanto mais próximo de 1, melhor é a predição.

Intervalo: [0, 1]

- 4) Precisão ou valor preditivo positivo (PPV):

$$Precision = \frac{TP}{TP + FP}$$

Mede a capacidade de um modelo não classificar amostras negativas como positivas. Denota a proporção de previsão de casos positivos (TP + FP) que são corretamente positivos reais (TP), ou seja, dentre todos os casos previstos como positivos, quantos são verdadeiramente positivos.

Objetivo: Quanto mais próximo de 1, melhor é a predição.

Intervalo: [0, 1]

- 5) Acurácia (Acc):

$$Acc = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

Mede a capacidade de acertar as previsões possíveis. Acurácia é o percentual de previsões que coincidem exatamente com os rótulos de classe verdadeiros, ou seja, é a razão entre o somatório das previsões corretas (TP + TN) sobre o somatório das previsões.

- 6) Pontuação F1 (média harmônica de *precision* e *recall*):

$$F_1 = 2 \cdot \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

A pontuação F1 é uma medida da acurácia de um modelo, combinando *precision* e *recall*. Uma boa pontuação F1 mais próxima de 1 indica poucos falsos positivos (FP) e poucos falsos negativos (FN). No entanto, não leva em conta os verdadeiros negativos.

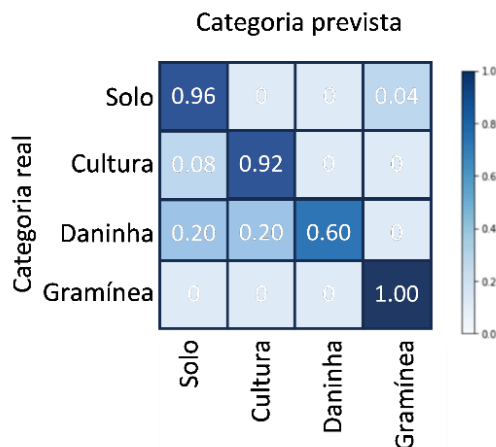
Objetivo: Quanto mais próximo de 1, melhor é a predição.

Intervalo: [0, 1]

Os cálculos acima analisam o modelo de classificação binária sobre os dados positivos, ou seja, se a instância pertence a uma classe real que estamos buscando prever,

portanto, as métricas avaliam esta classe. Em classificação multiclasse (mais de duas classes possíveis), cada classe é analisada separadamente.

Na matriz de confusão de um modelo de classificação multiclasse, consideramos a classe que queremos prever como a classe positiva, e todo o restante como negativo, assim, reduzimos para uma classificação binária para entender a capacidade do modelo em prever a classe que queremos. Na [Figura C.2](#), uma célula mais escura indica um número maior de amostras. A linha da matriz normalizada mostra a porcentagem da classe  $C_i$  prevista para ser a classe  $C_j$ . A matriz de confusão de um modelo bom terá a maioria das amostras ao longo da diagonal.



**Figura C.2** – Matriz de confusão de um modelo de classificação multiclasse.

As métricas usadas na multiclasse são as mesmas usadas no caso da classificação binária, mas exigem a média de classes para produzir uma pontuação para classificação multiclasse. A métrica é calculada para cada classe, tratando-a como um problema de classificação binária após agrupar todas as outras classes como pertencentes à segunda classe. Em seguida, a média da métrica binária é medida em todas as classes para obter uma métrica de média macro, micro ou ponderada.

- A média macro calcula a métrica para cada classe e calcula a média não ponderada.
- A média micro calcula a métrica globalmente contando o total de verdadeiros positivos, falsos negativos e falsos positivos;
- A média ponderada calcula a métrica para cada classe e calcula a média ponderada pelo número de amostras por classe (ponderada por frequência de classe).

Embora cada método de média tenha seus benefícios, uma consideração comum ao selecionar o método apropriado é o desequilíbrio de classe. Se as classes tiverem diferentes números de amostras, talvez seja mais informativo usar uma média de macro em que as classes minoritárias recebem peso igual ao das classes majoritárias [MICROSOFT].



Na classificação multiclasse, as nomenclaturas  $TP_c$ ,  $TF_c$ ,  $FP_c$  e  $FN_c$  indicam os números das classificações de verdadeiros positivos (detecções corretas), verdadeiro negativo, falsos positivos (detecções erradas) e falsos negativos (falha de detecção) da classe  $c$ , respectivamente.

A saída da rede de segmentação semântica com mapas de atributos de  $M \times N \times C$ , com  $C$  canais, mostra as probabilidades de cada *pixel* pertencer a cada classe definida. Por exemplo, com quatro classes  $[0, 1, 2, 3]$ , os elementos do canal 2 correspondem a probabilidades *pixel a pixel* de serem da classe  $C = 2$  (daninha).

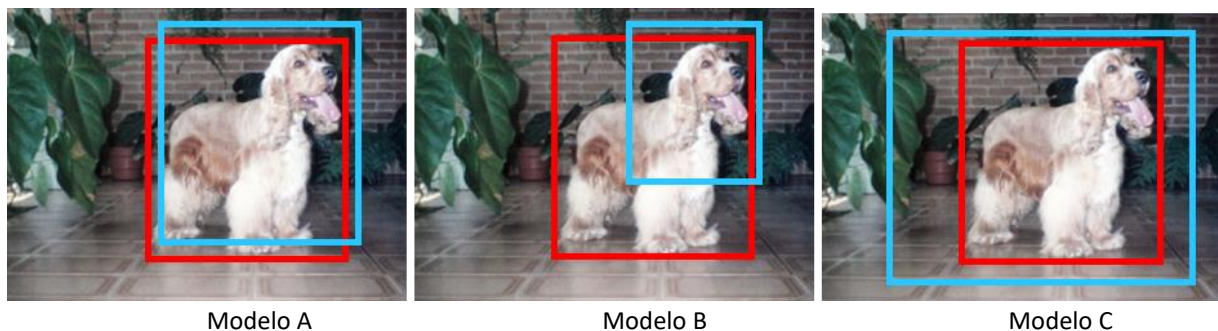
Para calcular  $TP_c$ ,  $TF_c$ ,  $FP_c$ ,  $FN_c$ , essas probabilidades de classe devem ser convertidas em valores binários dado um limite. No entanto, diferente do processo dos problemas de classificação binária, na segmentação semântica não é necessário escolher um limite de pontuação para fazer previsões. A resposta prevista é a classe com a maior pontuação prevista (*softmax*).

Os verdadeiros positivos ( $TP_2$ ) são as daninhas previstas corretamente como daninhas, os falsos positivos  $FP_2$  são os *pixels* de qualquer outra classe previstas erroneamente como daninhas e os falsos negativos  $FN_2$  são as falhas na detecção de daninhas, que são previstas como qualquer outra classe.

Com estes valores para cada classe é possível calcular a métrica de intersecção sobre união (IoU – *Intersection over Union*) sobre todas as classes, muito utilizada em segmentação de imagens.

## C.2 Intersecção sobre união para detecção de objetos

Considerando três modelos treinados para detectar cachorro, a [Figura C.3](#) mostra a caixa delimitadora de *groundtruth* (em vermelho) e as previsões dos modelos (em ciano).

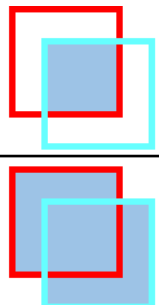


**Figura C.3** – *Detecção de objetos com caixa delimitadora.*

Comparando o modelo C com o modelo A, não é adequado determinar a área de sobreposição do *groundtruth* com a área prevista como medida de desempenho, sendo necessário uma métrica que penalize a medida sempre que (1) a previsão falha em prever a área dentro do *groundtruth* e (2) a previsão transborda o *groundtruth*.

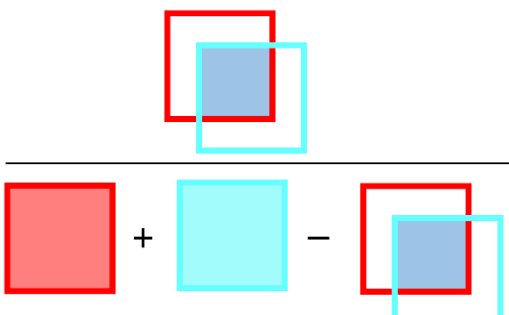
A métrica intersecção sobre união (IoU) corresponde à área de sobreposição entre a caixa prevista e o *groundtruth*, dividida pela área de união entre a previsão e o *groundtruth*, como mostra a [Figura C.4](#). O numerador é menor se a previsão falha em

prever a área dentro do *groundtruth*. Se a área da caixa prevista for maior, o denominador será maior, tornando o IoU menor.

$$IoU = \frac{\text{Área de sobreposição}}{\text{Área de união}} = \frac{\text{Área de sobreposição}}{\text{Área de sobreposição} + \text{Área de sobreposição} + \text{Área de sobreposição}}$$


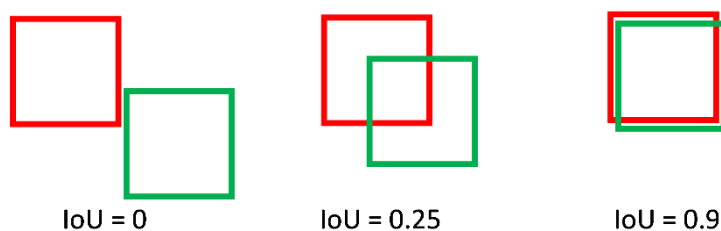
**Figura C.4** – *Intersecção sobre união.*

Na [Figura C.4](#), a área da interseção é adicionada duas vezes no denominador. Então, na verdade, IoU pode ser calculado conforme mostrado na [Figura C.5](#).

$$IoU = \frac{\text{Área de sobreposição}}{\text{Área de groundtruth} + \text{Área predita} - \text{Área de Sobreposição}}$$


**Figura C.5** – *Intersecção sobre união.*

Essa métrica varia de 0 a 1 (0 a 100%), onde 0 significa nenhuma sobreposição e 1 significa previsão perfeitamente sobreposta, como exemplificado na [Figura C.6](#).



**Figura C.6** – *Intersecção sobre união.*

### C.3 Intersecção sobre união para segmentação de imagens

Na segmentação de imagem, IoU é a métrica principal para avaliar a precisão do modelo. No caso da segmentação de imagem, a área pode ter qualquer forma irregular, ou seja, as previsões são máscaras de segmentação e não caixas delimitadoras (Figura C.7).

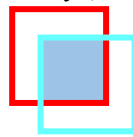


**Figura C.7** – Segmentação semântica de imagens: máscara de *groundtruth* em vermelho e máscara predita em ciano.

Agora temos:

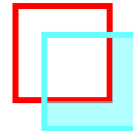
- Verdadeiro Positivo (TP): área de interseção entre *groundtruth* (GT) e máscara de segmentação (S), ou seja, é uma operação AND de GT e S,

$$TP = GT \cdot S$$



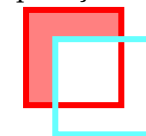
- Falso Positivo (FP): área prevista fora do *groundtruth*, ou seja, é a operação lógica OR de GT e S menos GT.

$$FP = (GT + S) - GT$$



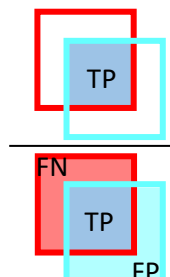
- Falso Negativo (FN): número de *pixels* na área do *groundtruth* que o modelo falhou em prever, ou seja, é a operação lógica OR de GT e S menos S.

$$FN = (GT + S) - S$$



A métrica IoU é a razão entre a área de interseção e a área de união de previsão e *groundtruth*, portanto, como os valores de TP, FP e FN nada mais são do que áreas ou número de *pixels*, o IoU pode ser escrito da seguinte forma:

$$IoU = \frac{TP}{(TP + FP + FN)}$$



# Apêndice D

## Informações adicionais

### D.1 Função de ativação Softmax

Para problemas de classificação usando técnicas de aprendizado profundo, é padrão usar a função de ativação *softmax* (também conhecida como regressão logística multinomial) na parte superior [TANG 2013]. Por exemplo, dadas  $n$  classes, a camada *softmax* tem  $n$  nós denotados por  $p_i$ , onde  $i = 0, \dots, n - 1$ . Cada nó  $p_i$  especifica uma distribuição de probabilidade discreta, portanto:

$$\sum_{i=0}^{n-1} p_i = 1$$

Seja  $h_k$  a ativação dos nós da penúltima camada, e  $W$  o peso que conecta a penúltima camada à camada *softmax*, a entrada total de uma unidade na camada *softmax*, dada por  $a_i$ , é

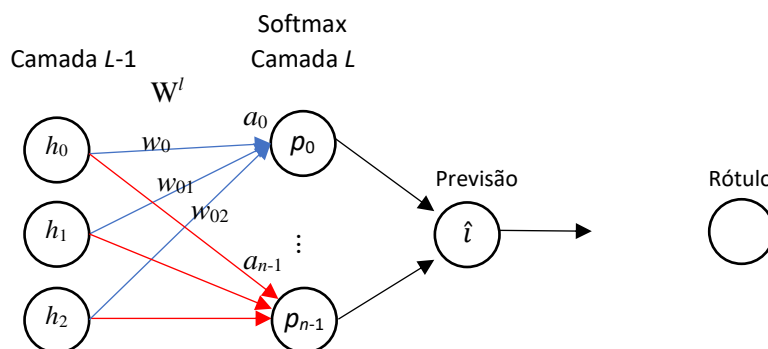
$$a_i = \sum_k h_k W_{ik}, \quad (\text{E.1})$$

então temos,

$$p_i = \frac{\exp(a_i)}{\sum_{j=0}^{n-1} \exp(a_j)} \quad (\text{E.2})$$

A classe prevista  $\hat{i}$  seria

$$\begin{aligned} \hat{i} &= \operatorname{argmax}_i p_i \\ &= \operatorname{argmax}_i a_i \end{aligned} \quad (\text{E.3})$$

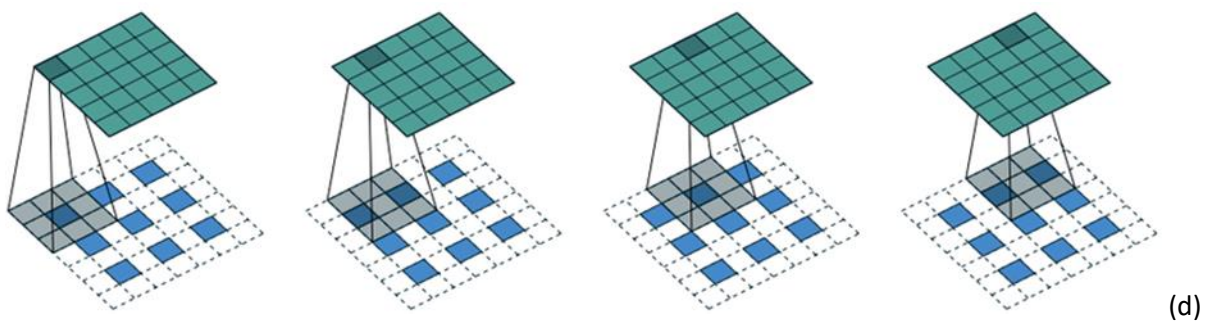
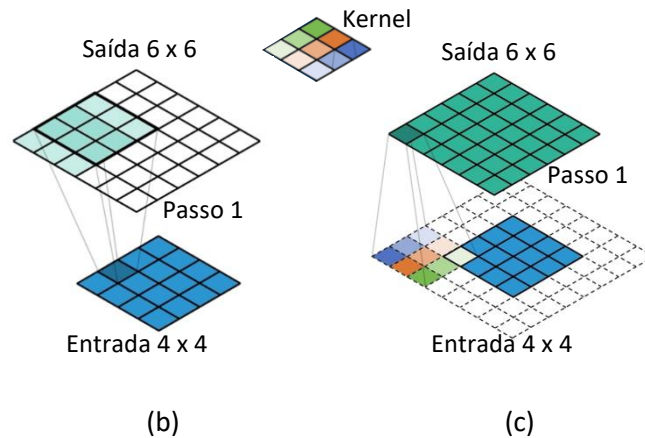
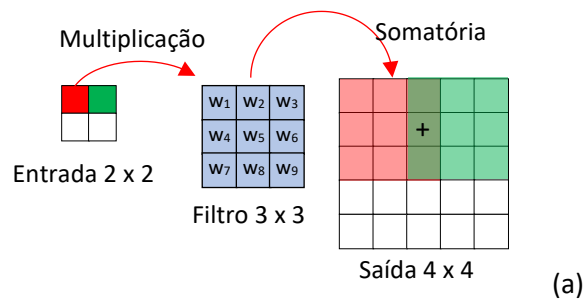


**Figura D.1** – Camada softmax das redes de classificação.

## D.2 Convolução transposta

As convoluções transpostas são usadas para fazer a sobreamostragem do mapa de atributos de entrada para um mapa de atributos de saída desejado usando alguns parâmetros que podem ser aprendidos. Cada elemento do mapa de atributos de entrada é multiplicado por cada peso do *kernel*, e as saídas sobrepostas são acumuladas, como mostra a [Figura D.2a](#) e [b](#). Na [Figura D.2c](#), a convolução transposta é implementada como uma convolução regular com filtro refletido.

A [Figura D.2d](#) mostra a convolução transposta para *upsampling* de 2x da entrada  $3 \times 3$  (azul) para uma saída  $5 \times 5$  (verde), com *kernel*  $3 \times 3$  e passo  $s = 2$  ( $s - 1 > 0$ : número de zeros inseridos entre elementos da entrada para obter sobreamostragem de  $s$  vezes). No entanto, esta operação foi implementada como convolução normal com *kernel*  $3 \times 3$  com passo  $s = 1$ .



**Figura D.2** – Convolução transposta [[DUMOULIN e VISIN 2016](#)].



# Referências bibliográficas

- [ABADI *et al.* 2015] Martin ABADI *et al.* TensorFlow: large-scale machine learning on heterogeneous distributed systems. 2015. URL: <https://www.tensorflow.org/>.
- [ABU-MOSTAFA *et al.* 2012] Yaser S. ABU-MOSTAFA, Malik MAGDON-ISMAIL e Hsuan-Tien LIN. *Learning from data*. vol. 4. AMLBook New York, NY, USA: 2012.
- [ACHANTA *et al.* 2012] Radhakrishna ACHANTA, Appu SHAJI, Kevin SMITH, Aurélien LUCCHI, Pascal FUA e Sabine SÜSTRUNK. “SLIC Superpixels compared to state-of-the-art superpixel methods”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, n. 11, pgs. 2274–2282, 2012. URL: [pdf](#).
- [BADRINARAYANAN *et al.* 2015] Vijay BADRINARAYANAN, Ankur HANDA e Roberto CIPOLLA. “SegNet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling”, 2015. arXiv: [1505.07293](#).
- [BADRINARAYANAN *et al.* 2017] Vijay BADRINARAYANAN, Alex KENDALL e Roberto CIPOLLA. “SegNet: a deep convolutional encoder-decoder architecture for image segmentation”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, n. 12, pgs. 2481-2495, 2017. DOI: [10.1109/TPAMI.2016.2644615](#). arXiv: [1511.00561](#).
- [BARNES *et al.* 2000] E. M. BARNES, T. R. CLARKE, S. E. RICHARDS, P. D. COLAIZZI, J. HABERLAND, M. KOSTRZEWSKI, P. WALLER, C. CHOI, E. RILEY, T. THOMPSON, R. J. LASCANO, H. LI e M.S. MORAN. “Coincident detection of crop water stress, nitrogen status, and canopy density using ground-based multispectral data”. Em: *5th International Conference on Precision Agriculture*, Bloomington, 16-19 July 2000, pgs. 1-15, 2000.
- [BERG *et al.* 2010] Alex BERG, Jia DENG e Li FEI-FEI. Large Scale Visual Recognition Challenge (ILSVRC-2010), 2010. URL: <http://www.image-net.org/challenges/LSVRC/2010/>.
- [BIRTH 1968] G. S. BIRTH, G. MCVEY. “Measuring the colour of growing turf with a reflectance spectrophotometer”. *Agronomy Journal*, vol. 60, n. 6, pgs. 640-643, 1968. <http://dx.doi.org/10.2134/agronj1968.00021962006000060016x>
- [BREIMAN 2001] Leo BREIMAN. “Random forests”. *Machine Learning*, vol. 45, pgs. 5–32, 2001. DOI: [10.1023/A:1010933404324](#).
- [BRIGHENTI 2010] Alexandre M. BRIGHENTI. *Manual de identificação e manejo de plantas daninhas em cultivos de cana-de-açúcar*. Livro técnico, Embrapa Gado de Leite, Juiz de Fora, 2010.
- [BRITO e COELHO 2012] Jorge N. BRITO e Luiz COELHO. *Fotogrametria digital*, Instituto Militar de Engenharia, 1ª ed., Rio de Janeiro, Brasil, 2002.
- [BROSTOW *et al.* 2009] Gabriel BROSTOW, Julien FAUQUEUR e Roberto CIPOLLA. “Semantic object classes in video: a high-definition ground truth database”, *PRL*, vol. 30, n. 2, pgs. 88-97, 2009. URL: [pdf](#).

- [BROWN e LOWE 2007] Matthew BROWN, David G. LOWE. “Automatic Panoramic Image Stitching using Invariant Features”. *International Journal of Computer Vision*, vol. 74, n. 1, pgs. 59-73, 2007. URL: [pdf](#).
- [BRYSON *et al.* 2010] Mitch BRYSON, Alistair REID, Fabio RAMOS e Salah SUKKARIEH. “Airborne vision-based mapping and classification of large farmland environments”. Em: *Journal of Field Robotics*, vol. 27, n. 5, pgs. 632-655, 2010. URL: [pdf](#).
- [BURT e ADELSON 1983] P. J. BURT e E. H. ADELSON. “The laplacian pyramid as a compact image code”. Em: *IEEE Transactions on Communications*, vol. 31, n.4, pgs. 532-540, 1983. DOI: [10.1109/TCOM.1983.1095851](#).
- [CAESAR *et al.* 2016] Holger CAESAR, Jasper UIJLINGS e Vittorio FERRARI. “COCO-Stuff: thing and stuff classes in context”. arXiv: [1612.03716](#), 2016.
- [CÉSAR PEREIRA *et al.* 2019] Paulo CÉSAR PEREIRA Júnior, Alexandre MONTEIRO e Aldo Von WANGENHEIM. Weed Mapping on Aerial Images - A Systematic Literature Review. Relatório técnico INCoD/LAPIX.01.2019.E (Maio, 2019). DOI: [10.13140/RG.2.2.34979.71204](#). URL: <https://lapix.ufsc.br/weed-mapping-sugar-cane/>
- [CHEN *et al.* 2014] Xianjie CHEN, Roozbeh MOTTAGHI, Xiaobai LIU, Sanja FIDLER, Raquel URTASUN e Alan YUILLE, “Detect what you can: detecting and representing objects using holistic models and body parts”. Em: *CVPR*, 2014. arXiv: [1406.2031](#).
- [CHEN *et al.* 2015] Liang-Chieh CHEN, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY e Alan L. YUILLE. “Semantic image segmentation with deep convolutional nets and fully connected CRFs”. Em: *Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA*, 2015. arXiv: [1412.7062](#).
- [CHEN *et al.* 2016a] Liang-Chieh CHEN, Yi YANG, Jiang WANG, Wei XU e Alan L. YUILLE. “Attention to scale: scale-aware semantic image segmentation”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, pgs. 3640-3649, 2016. DOI: [10.1109/CVPR.2016.396](#). arXiv: [1511.03339](#).
- [CHEN *et al.* 2017] Liang-Chieh CHEN, George PAPANDREOU, Florian SCHROFF e Hartwig ADAM. “Rethinking atrous convolution for semantic image segmentation”, 2017. arXiv: [1706.05587](#).
- [CHEN *et al.* 2018a] Liang-Chieh CHEN, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY e Alan L. YUILLE. “DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, n. 4, pgs. 834-848, 2018. DOI: [10.1109/TPAMI.2017.2699184](#). arXiv: [1606.00915](#).
- [CHEN *et al.* 2018b] Liang-Chieh CHEN, Yukun ZHU, George PAPANDREOU, Florian SCHROFF e Hartwig ADAM. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. Em: *Proceedings of the 15th European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science, 11211, pgs. 833-851, 2018. DOI: [10.1007/978-3-030-01234-2\\_49](#).
- [CHOLLET 2016] François CHOLLET. *Deep learning with Python*. New York, NY, USA: Manning Publications Co, 2017. ISBN:978-1-61729-443-3.
- [CHOLLET 2017] François CHOLLET. “Xception: deep learning with depthwise separable convolutions”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, pgs. 1800-1807, 2017. DOI: [10.1109/CVPR.2017.195](#). arXiv: [1610.02357](#).
- [CHOLLET 2018] François CHOLLET. *Keras machine learning framework*. 2020. URL: <https://github.com/fchollet/keras>.

- [CORDTS *et al.* 2016] Marius CORDTS, Mohammed OMRAN, Sebastian RAMOS, Timo REHFELD, Markus ENZWEILER, Rodrigo BENENSON, Uwe FRANKE, Stefan ROTH e Bernt SCHIELE. “The Cityscapes dataset for semantic urban scene understanding”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, pgs. 3213-3223, 2016. DOI: [10.1109/CVPR.2016.350](https://doi.org/10.1109/CVPR.2016.350). arXiv: [1604.01685](https://arxiv.org/abs/1604.01685).
- [CORTES e VAPNIK 1995] Corinna CORTES e Vladimir VAPNIK. "Support-vector networks". *Machine Learning*. vol. 20, pgs. 273–297, 1995. DOI: [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).
- [COVER *et al.* 1967] Thomas M. COVER e Peter E. HART. “Nearest neighbor pattern classification”. *IEEE Transactions on Information Theory*, vol. 13, pgs. 21–27, 1967. CiteSeerX [10.1.1.68.2616](https://citeseerx.ist.psu.edu/viewdoc/doi?doi=10.1.1.68.2616). DOI: [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964). URL: [pdf](#).
- [DENG *et al.* 2009] Jia DENG, Wei DONG, Richard SOCHER, Li-Jia LI, Kai LI e Li FEI-FEI. “ImageNet: a large-scale hierarchical image database”. Em: 2009 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pgs. 248-255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848). URL: [pdf](#).
- [DENG *et al.* 2012] Jia DENG, Alex BERG, Sanjeev SATHEESH, Hao SU, Aditya KHOSLA e Li FEI-FEI. ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012), 2012. URL: <http://www.image-net.org/challenges/LSVRC/2012/>.
- [DROZDZAL *et al.* 2016] Michal DROZDZAL, Eugene VORONTSOV, Gabriel CHARTRAND, Samuel KADOURY e Chris PAL. “The importance of skip connections in biomedical image segmentation”. arXiv: [1608.04117](https://arxiv.org/abs/1608.04117), 2016.
- [DUMOULIN e VISIN 2016] Vincent DUMOULIN e Francesco VISIN. “A guide to convolution arithmetic for deep learning”, 2016, arXiv: [1603.07285](https://arxiv.org/abs/1603.07285).
- [EVERINGHAM *et al.* 2015] Mark EVERINGHAM, S. M. Ali ESLAMI, Luc Van GOOL, Christopher K. I. WILLIAMS, John WINN e Andrew ZISSERMAN. “The PASCAL visual object classes challenge: a retrospective”. Em: *International Journal of Computer Vision (IJCV)*, vol. 111, n. 1, pgs. 98-136, 2015. DOI: [10.1007/s11263-014-0733-5](https://doi.org/10.1007/s11263-014-0733-5).
- [FARABET *et al.* 2013] C. FARABET, C. COUPRIE, L. NAJMAN e Y. LECUN. “Learning hierarchical features for scene labeling”. *Pattern Analysis and Machine Intelligence*, IEEE Transactions on, 2013.
- [FERREIRA *et al.* 2017] Alessandro dos Santos FERREIRA, Daniel Matte FREITAS, Gercina Gonçalves da SILVA, Hemerson PISTORI e Marcelo Theophilo FOLHES. “Weed detection in soybean crops using ConvNets”. Em: *Computers and Electronics in Agriculture*, vol. 143, pgs. 314-324, 2017. DOI: [10.1016/j.compag.2017.10.027](https://doi.org/10.1016/j.compag.2017.10.027).
- [FILHO e CHRISTOFFOLETI 1987] Ricardo Victoria FILHO e Pedro Jacob CHRISTOFFOLETI. “Manejo de plantas daninhas e produtividade da cana”. *Visão Agrícola*, pgs. 2-9, 1987. URL: [pdf](#).
- [FISCHLER e BOLLES 1981] M. A. FISCHLER e R. C. BOLLES. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. *Communications of the ACM*, 24, vol. 6, pgs. 381–395, 1981.
- [FLETCHER *et al.* 2016] Reginald S. FLETCHER. “Using vegetation indices as input into random forest for soybean and weed classification”. *American Journal of Plant Sciences*, vol. 7, n. 15, pgs. 2186-2198, 2016. DOI: [10.4236/ajps.2016.715193](https://doi.org/10.4236/ajps.2016.715193).
- [FURUKAWA e HERNÁNDEZ 2015] Yasutaka FURUKAWA e Carlos HERNÁNDEZ. “Multi-view stereo: A tutorial”. *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, n. 1-2, pgs. 1-148, 2015.
- [GAO *et al.* 2018] Junfeng GAO, Wenzhi LIAO, David NUYTENS, Peter LOOTENS, Jürgen VANGHEYTE, Aleksandra PIŽURICA, Yong HE e Jan G. PIETERS. “Fusion of pixel and

- object-based features for weed mapping using unmanned aerial vehicle imagery”. Em: *International Journal of Applied Earth Observation and Geoinformation*, vol. 67, pgs. 43-53, 2018. DOI: [10.1016/j.jag.2017.12.012](https://doi.org/10.1016/j.jag.2017.12.012).
- [GEIGER *et al.* 2012] Andreas GEIGER, Philip LENZ e Raquel URTASUN. “Are we ready for autonomous driving? the KITTI vision benchmark suite”. Em: *CVPR*, pgs. 3354-3361, 2012. DOI: [10.1109/CVPR.2012.6248074](https://doi.org/10.1109/CVPR.2012.6248074). URL: [pdf](#).
- [GÉRON 2019] Aurélien GÉRON. *Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow: concepts, tools, and techniques to build intelligent systems*. 2<sup>a</sup> ed. [S.l.]: O' Reilly Media, 2019. ISBN: 9781492032649.
- [GHIASI e FOWLKES 2016] Golnaz GHIASI e Charless C. FOWLKES. “Laplacian pyramid reconstruction and refinement for semantic segmentation”, 2016. arXiv: [1605.02264](https://arxiv.org/abs/1605.02264).
- [GIRSHICK *et al.* 2014] Ross GIRSHICK, Jeff DONAHUE, Trevor DARRELL e Jitendra MALIK. “Rich feature hierarchies for accurate object detection and semantic segmentation”. Em: *Computer Vision and Pattern Recognition*, 2014. URL: [pdf](#).
- [GITELSON *et al.* 1996] Anatoly A. GITELSON, Yoram J. KAUFMAN e Mark N. MERZLYAK. “Use of a green channel in remote sensing of global vegetation from EOS-MODIS”. *Remote Sensing of Environment*, vol. 58, pgs. 289-298, 1996. DOI: [10.1016/S0034-4257\(96\)00072-7](https://doi.org/10.1016/S0034-4257(96)00072-7).
- [GLOROT e BENGIO 2010] Xavier GLOROT e Yoshua BENGIO. “Understanding the difficulty of training deep feedforward neural networks”. Em: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pgs. 249-256, 2010. URL: [pdf](#).
- [GOODFELLOW *et al.* 2016] Ian J. GOODFELLOW, Yoshua BENGIO e Aaron C. COURVILLE. *Deep learning: adaptive computation and machine learning*. MIT Press, 2016. ISBN: 978-0-262-03561-3. URL: <http://www.deeplearningbook.org/>.
- [GUERRERO *et al.* 2012] J. M. GUERRERO, G. PAJARES, M. MONTALVO, J. ROMEO e M. GUIJARRO. “Support Vector Machines for crop/weeds identification in maize Fields”. *Expert Syst. Appl.* 2012, vol. 39, pgs. 11149–11155. DOI: [10.1016/j.eswa.2012.03.040](https://doi.org/10.1016/j.eswa.2012.03.040).
- [GUO *et al.* 2019] Yahui GUO, J. SENTHILNATH, Wenxiang WU, Xueqin ZHANG, Zhaoqi ZENG e Han HUANG. “Radiometric calibration for multispectral camera of different imaging conditions mounted on a UAV platform”. *Sustainability* (Switzerland), vol. 11, n. 4, 2019.
- [HABOUDANE *et al.* 2004] Driss HABOUDANE, John R. MILLER, Elizabeth PATTEY, Pablo J. ZARCO-TEJADA e Iam B. STRACHAN. “Hyperspectral vegetation indices and novel algorithms for predicting green LAI of crop canopies: modeling and validation in the context of precision agriculture”. *Remote Sensing of Environment*, vol. 90, n. 3, pgs. 337-352, 2004. DOI: [10.1016/j.rse.2003.12.013](https://doi.org/10.1016/j.rse.2003.12.013).
- [HARIHARAN *et al.* 2014] Bharath HARIHARAN, Pablo ARBELAEZ, Ross GIRSHICK e Jitendra MALIK. “Simultaneous detection and segmentation”. Em: *European Conference on Computer Vision (ECCV)*, 2014, arXiv: [1407.1808](https://arxiv.org/abs/1407.1808).
- [HARIHARAN *et al.* 2014a] Bharath HARIHARAN, Pablo ARBELAEZ, Ross GIRSHICK e JITENDRA MALIK. “Hypercolumns for object segmentation and fine-grained localization”. arXiv: [1411.5752](https://arxiv.org/abs/1411.5752), 2014.
- [HARTLEY e ZISSERMAN 2003] Richard HARTLEY e Andrew ZISSERMAN. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. ISBN 978-0-521-54051-3.



- [HATFIELD *et al.* 2008] Jerry L. HATFIELD, Anatoly A. GITELSON, J. S. SCHEPERS e Charlie L. WALTHALL. “Application of spectral remote sensing for agronomic decisions”. *Agronomy Journal*, vol. 100, pgs. 117- 131, 2008. Supplement to *Agronomy Journal: Celebrate the Centennial*. URL: [pdf](#).
- [HE *et al.* 2015b] Kaiming HE, Xiangyu ZHANG, Shaoqing REN e Jian SUN. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, n. 9, pgs. 1904-1916, 2015. DOI: [10.1109/TPAMI.2015.2389824](#). arXiv: [1406.4729](#).
- [HE *et al.* 2016a] Kaiming HE, Xiangyu ZHANG, Shaoqing REN e Jian SUN. “Deep residual learning for image recognition”. Em: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pgs. 770-778. DOI: [10.1109/CVPR.2016.90](#). arXiv: [1512.03385](#).
- [HE *et al.* 2016b] Kaiming HE, Xiangyu ZHANG, Shaoqing REN e Jian SUN. “Identity mappings in deep residual networks”. Em: *European Conference on Computer Vision (ECCV), 2016*. Lecture Notes in Computer Science, vol. 9908, pgs. 630-645, 2016. DOI: [10.1007/978-3-319-46493-0\\_38](#).
- [HINTON *et al.* 2012] Geoffrey E. HINTON, Nitish SRIVASTAVA, Alex KRIZHEVSKY, Ilya SUTSKEVER e Ruslan R. SALAKHUTDINOV. “Improving neural networks by preventing co-adaptation of feature detectors”, 2012. arXiv: [1207.0580](#).
- [HOLLER *et al.* 2022] Wilson Anderson HOLLER, Bruna Nascimento de VASCONCELLOS, Bruno Holtz GEMIGNANI, Ana Paula Dalla CORTE, Adriane Avelhaneda MALLMANN. *Câmeras modificadas e multiespectrais embarcadas em drones: enfoque para estudos da vegetação*. Embrapa Florestas, vol. 29, pgs.1980-3958, 2022. URL: [pdf](#).
- [HOWARD 2014] Andrew G. HOWARD. “Some improvements on deep convolutional neural network based image classification”. Em: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2014. arXiv: [1312.5402](#).
- [HOWARD *et al.* 2017] Andrew G. HOWARD, Menglong ZHU, Bo CHEN, Dmitry KALENICHENKO, Weijun WANG, Tobias WEYAND, Marco ANDREETTO e Hartwig ADAM. “MobileNets: efficient convolutional neural networks for mobile vision applications”, 2017. arXiv: [1704.04861](#).
- [HUANG *et al.* 2017] Gao HUANG, Zhuang LIU, Laurens van der MAATEN e Kilian Q. WEINBERGER. “Densely connected convolutional networks”. Em: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pgs. 2261-2269, 2017. DOI: [10.1109/CVPR.2017.243](#). arXiv: [1608.06993](#).
- [HUANG *et al.* 2018] Huasheng HUANG, Jizhong DENG, Yubin LAN, Aqing YANG, Xiaoling DENG e Lei ZHANG. “A fully convolutional network for weed mapping of unmanned aerial vehicle (UAV) imagery”. *PLoS ONE*, vol. 13, n. 4, e0196302, 2018. DOI: [10.1371/journal.pone.0196302](#). URL: [pdf](#).
- [HUNG *et al.* 2014] Calvin HUNG, Zhe XU e Salah SUKKARIEH. “Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV”. *Remote Sensing*, vol. 6, n. 12, pgs. 12037-12054, 2014. DOI: [10.3390/rs61212037](#).
- [INAMASU *et al.* 2011] Ricardo Y. INAMASU, Aberto C. de Campos BERNARDI, Carlos Manoel Pedro VAZ, João de Mendonça NAIME, Leonardo Ribeiro QUEIROS, Álvaro Vilela de RESENDE, Marina de Fátima VILELA, Lúcio André de Castro JORGE, Luís Henrique BASSOI, Naylor Bastiani PEREZ, Edilson Pepino FRAGALLE. “Agricultura de precisão para a sustentabilidade de sistemas produtivos do agronegócio brasileiro”. In: INAMASU, R. Y.; NAIME, J. M.; RESENDE, A. V.;



- BASSOI, L. H.; BERNARDI, A. C. C. (Ed.). Agricultura de precisão: um novo olhar. São Carlos: Embrapa Instrumentação, 2011. pgs. 14-26. URL: [pdf](#).
- [IOFFE e SZEGEDY 2015] Sergey IOFFE e Christian SZEGEDY. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. Em: *Proceedings of the 32nd International Conference Machine Learning (ICML), Lille, France, 37* (2015), pgs. 448-456. arXiv: [1502.03167](#).
- [JÉGOU *et al.* 2017] Simon JÉGOU, Michal DROZDZAL, David VAZQUEZ, Adriana ROMERO e Yoshua BENGIO. “The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation”. Em: *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017. IEEE Conference on, pgs. 1175-1183. IEEE, 2017. arXiv: [1611.09326](#).
- [JENSEN 2009] John R. JENSEN. *Sensoriamento Remoto do Ambiente: Uma Perspectiva em Recursos Terrestres*, 2009.
- [JIN *et al.* 2020] Xin JIN, Cuiling LAN, Wenjun ZENG, Zhizheng ZHANG e Zhibo CHEN. “CASINet: content-adaptive scale interaction networks for scene parsing”, 2020. arXiv: [1904.08170](#).
- [JORGE e INAMASU 2014] Lúcio André de Castro JORGE e Ricardo Y. INAMASU. “Uso de veículos aéreos não tripulados (VANT) em Agricultura de Precisão” Em: A. C. C. BERNARDI, J. M. NAIME, A. V. RESENDE, L. H. BASSOI, R. Y. INAMASU (Ed.). Agricultura de precisão: resultados de um novo olhar. Brasília, DF: Embrapa, 2014. pgs. 109-134. URL: [pdf](#).
- [KAMILARIS e PRENAFETA-BOLDÚ 2018] Andreas KAMILARIS e Francesc X. PRENAFETA-BOLDÚ. “A review of the use of convolutional neural networks in agriculture”. Em: *The Journal of Agricultural Science*, vol. 156, n. 3, pgs. 312-322, 2018. DOI: [10.1017/S0021859618000436](#).
- [KENDALL *et al.* 2015] Alex KENDALL, Vijay BADRINARAYANAN e Roberto CIPOLLA. “Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding”. arXiv: [1511.02680](#), 2015.
- [KESKAR *et al.* 2016] Nitish S. KESKAR, Jorge NOCEDAL, Ping Tak Peter TANG, Dheevatsa MUDIGERE e Mikhail SMELYANSKIY. “On large-batch training for deep learning: Generalization GAP and sharp minima”. arXiv: [1609.04836](#), 2016.
- [KINGMA e BA 2014] Diederik P. KINGMA e Jimmy Lei BA. “Adam: a method for stochastic optimization”. arXiv: [1412.6980](#), 2014.
- [KRAHENBUHL e KOLTUN 2011] Philipp KRAHENBUHL e Vladlen KOLTUN. “Efficient inference in fully connected crfs with gaussian edge potentials”. Em: *NIPS*, 2011. arXiv: [1210.5644](#).
- [KRIZHEVSKY *et al.* 2012] Alex KRIZHEVSKY, Ilya SUTSKEVER e Geoffrey E. HINTON. “ImageNet classification with deep convolutional neural networks”. Em: *Advances in Neural Information Processing Systems 25 (NIPS)*. 2012, pgs. 1097-1105. URL: [pdf](#).
- [LECUN 1989] Yann LECUN. “Generalization and network design strategies”. Em: *Connectionism in Perspective*, R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, eds North-Holland, Amsterdam. 1989.
- [LECUN *et al.* 1989] Yann LECUN, Bernhard E. BOSER, John S. DENKER, Donnie HENDERSON, Richard E. HOWARD, Wayne HUBBARD e Lawrence D. JACKEL. “Backpropagation applied to handwritten zip code recognition”. Em: *Neural Computation*, 1.4 (1989), pgs. 541-551. DOI: [10.1162/neco.1989.1.4.541](#).
- [LECUN *et al.* 1990] Yann LECUN, Bernhard E. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD e L. D. JACKEL. “Handwritten digit recognition with a

back-propagation network”. Em: *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. Ed. por David S. TOURETZKY. Morgan Kaufmann, 1989, pgs. 396-404. URL: [pdf](#).

- [LECUN *et al.* 1998] Yann LECUN, Léon BOTTOU, Yoshua BENGIO e Patrick HAFFNER. “Gradient-based learning applied to document recognition”. Em: *Proceedings of the IEEE*, 86.11 (1998), pgs. 2278-2324. DOI: [10.1109/5.726791](#). URL: [pdf](#).
- [LECUN *et al.* 2012] Yann LECUN, Léon BOTTOU, Genevieve ORR e Klaus R. MÜLLER. *Efficient BackProp*. Neural networks: tricks of the trade (2nd ed.), vol. 7700 de Lecture Notes in Computer Science, Springer, pgs. 9-48, 2012. URL: [pdf](#).
- [LECUN *et al.* 2015] Yann LECUN, Yoshua BENGIO e Geoffrey E. HINTON. “Deep learning”. Em: *Nature*, 521.7553 (2015), pgs. 436-444. DOI: [10.1038/nature14539](#).
- [LI *et al.* 2017a] Fei-Fei LI, Justin JOHNSON e Serena YEUNG. “Lecture 11: Detection and Segmentation. Cs231n: convolutional neural networks for visual recognition”. Em: *University lecture* (2017). URL: [pdf](#).
- [LI *et al.* 2018] Hanchao LI, Pengfei XIONG, Jie AN e Lingxue WANG. “Pyramid attention network for semantic segmentation”, 2018. arXiv: [1805.10180](#).
- [LIN *et al.* 2016] Guosheng LIN, Chunhua SHEN, Anton Van Den HENGEL e Ian REID. “Efficient piecewise training of deep structured models for semantic segmentation”. Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pgs. 3194–3203, 2016. arXiv: [1504.01013](#).
- [LIN, MAIRE *et al.* 2014] Tsung-Yi LIN, Michael MAIRE, Serge BELONGIE, James HAYS, Pietro PERONA, Deva RAMANAN, Piotr DOLLAR e C. Lawrence ZITNICK. “Microsoft COCO: Common Objects in Context”. Em: *Computer Vision-European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science, 8693 (2014), pgs. 740-755. URL: <https://cocodataset.org/>. arXiv: [1405.0312](#).
- [LIN, MILAN *et al.* 2017] Guosheng LIN, Anton MILAN, Chunhua SHEN e Ian D. REID. “RefineNet: multi-path refinement networks for high-resolution semantic segmentation”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, pgs. 5168-5177, 2017. DOI: [10.1109/CVPR.2017.549](#). arXiv: [1611.06612](#).
- [LIU *et al.* 2015] Wei LIU, Andrew RABINOVICH e Alexander C. BERG. “ParseNet: looking wider to see better”, 2015. arXiv: [1506.04579](#).
- [LONG *et al.* 2015] Jonathan LONG, Evan SHELHAMER e Trevor DARRELL. “Fully convolutional networks for semantic segmentation”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA*, pgs. 3431-3440, 2015. DOI: [10.1109/CVPR.2015.7298965](#). arXiv: [1411.4038](#).
- [LOTES *et al.* 2017] Philipp LOTES, Raghav KHANNA, Johannes PFEIFER, Roland SIEGWART e Cyrill STACHNISS. “UAV-based crop and weed classification for smart farming”. Em: *IEEE International Conference on Robotics and Automation (ICRA)*, pgs. 3024 - 3031, Singapore, 2017. DOI: [10.1109/ICRA.2017.7989347](#).
- [LOWE 2004] David G. LOWE. “Distinctive image features from scale-invariant keypoints”. *International Journal of Computer Vision*, vol. 60, n. 2, pgs. 91-110, 2004. URL: [pdf](#).
- [LU 2019] YANG LU. Código-fonte de referência das redes de segmentação semântica. URL: <https://github.com/luyanger1799/Amazing-Semantic-Segmentation>.
- [LUNA e LOBO 2016] Inti LUNA e Agustín LOBO. “Mapping crop planting quality in sugarcane from UAV imagery: a pilot study in Nicaragua”. *Remote Sensing*, vol. 8, n. 6, pgs. 500, 2016. DOI: [10.3390/rs8060500](#).

- [MALLAT 2008] Stéphane MALLAT. *A wavelet tour of signal processing: the sparse way*. 3<sup>a</sup> ed. [S.l.]: Academic Press, 2008. ISBN 13: 978-0-12-374370-1.
- [MATTIVI *et al.* 2021] Pietro MATTIVI, Salvatore E. PAPPALARDO, Nebojša NIKOLIĆ, Luca MANDOLESI, Antonio PERSICHETTI, Massimo De MARCHI, Roberta MASIN. “Can Commercial Low-Cost Drones and Open-Source GIS Technologies Be Suitable for Semi-Automatic Weed Mapping for Smart Farming? A Case Study in NE Italy”. *Remote Sens.* 2021, vol. 13, 1869. DOI: [10.3390/rs13101869](https://doi.org/10.3390/rs13101869).
- [MEINGAST *et al.* 2005] Marci MEINGAST, Christopher GEYER e Shankar SASTRY. “Geometric models of rolling-shutter cameras”. arXiv 2005, arXiv:cs/0503076. URL: [pdf](#).
- [MICROSOFT] Avaliar os resultados do experimento de machine learning automatizado. Acessado em julho de 2023. URL: <https://learn.microsoft.com/pt-br/azure/machine-learning/how-to-understand-automated-ml?view=azureml-api-2>.
- [MONTEIRO 2019] Alexandre Alvarenga de Oliveira MONTEIRO, “Segmentação e Identificação de Cultivo e Espécies Invasivas em Imagens Aéreas utilizando redes neurais convolucionais”, Dissertação de Mestrado, Universidade Federal de Santa Catarina -UFSC, 2019. URL: <https://repositorio.ufsc.br/handle/123456789/215694>
- [MONTEIRO e WANGENHEIM 2019] Alexandre MONTEIRO e Aldo von WANGENHEIM. Orthomosaic Dataset of RGB aerial Images for Weed Mapping, 2019. URL: <http://www.lapix.ufsc.br/weed-mapping-sugar-cane>.
- [MOTTAGHI *et al.* 2014] Roozbeh MOTTAGHI, Xianjie CHEN, Xiaobai LIU, Nam-Gyu CHO, Seong-Whan LEE, Sanja FIDLER, Raquel URTASUN e Alan YUILLE, “The role of context for object detection and semantic segmentation in the wild”. Em: *CVPR*, 2014. DOI: [10.1109/CVPR.2014.119](https://doi.org/10.1109/CVPR.2014.119). URL: [pdf](#).
- [NAIR e HINTON 2010] Vinod NAIR e Geoffrey E. HINTON. “Rectified linear units improve restricted Boltzmann machines”. Em: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pgs. 807-814. URL: [pdf](#).
- [NIELSEN 2015] Michael A. NIELSEN. *Neural networks and deep learning*. [S.l.]: Determination Press, 2015. URL: <http://neuralnetworksanddeeplearning.com/>.
- [NOGUEIRA *et al.* 2016] Keiller NOGUEIRA, Mauro Dalla MURA, Jocelyn CHANUSSOT, William R. SCHWARTZ e Jefersson A. dos SANTOS. “Learning to semantically segment high-resolution remote sensing images”. Em: *23rd International Conference on Pattern Recognition (ICPR), Cancun*, pgs. 3566-3571, 2016. DOI: [10.1109/ICPR.2016.7900187](https://doi.org/10.1109/ICPR.2016.7900187). URL: [pdf](#).
- [NOH *et al.* 2015] Hyeonwoo NOH, Seunghoon HONG e Bohyung HAN. “Learning deconvolution network for semantic segmentation”. Em: *IEEE International Conference on Computer Vision (ICCV), Santiago, Chile*, pgs. 1520-1528, 2015. DOI: [10.1109/ICCV.2015.178](https://doi.org/10.1109/ICCV.2015.178). arXiv: [1505.04366](https://arxiv.org/abs/1505.04366).
- [OPENDRONEMAP] OpenDroneMap Authors ODM – A command line toolkit to generate maps, point clouds, 3D models and DEMs from drone, balloon or kite images. OpenDroneMap/ODM      GitHub      Page      2020;      URL: <https://github.com/OpenDroneMap/ODM>
- [OQUAB *et al.* 2014] Maxime OQUAB, Léon BOTTOU, Ivan LAPTEV e Josef SIVIC. “Learning and transferring mid-level image representations using convolutional neural networks”. Em: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pgs. 1717-1724. DOI: [10.1109/CVPR.2014.222](https://doi.org/10.1109/CVPR.2014.222). URL: [pdf](#).

- [OTSU 1979] Nobuyuki A. OTSU. “Threshold Selection Method from Gray-Level Histograms”. Em: *IEEE Trans. Syst. Man Cybern.* 1979, vol. 9, pgs. 62–66. DOI: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076).
- [PASZKE *et al.* 2016] Adam PASZKE, Abhishek CHAURASIA, Sangpil KIM e Eugenio CULURCIELLO. “ENet: a deep neural network architecture for real-time semantic segmentation”, 2016. arXiv: [1606.02147](https://arxiv.org/abs/1606.02147).
- [PEÑA *et al.* 2015] José M. PEÑA, Jorge TORRES-SÁNCHEZ, Angélica SERRANO-PÉREZ, Ana I. de CASTRO e Francisca LÓPEZ-GRANADOS. “Quantifying efficacy and limits of unmanned aerial vehicle (UAV) technology for weed seedling detection as affected by sensor resolution”. *Sensors*, vol. 15, n. 3, pgs. 5609-5626, 2015. DOI: [10.3390/s150305609](https://doi.org/10.3390/s150305609).
- [PENG *et al.* 2017] Chao PENG, Xiangyu ZHANG, Gang Yu, Guiming LUO e Jian SUN. “Large kernel matters – improve semantic segmentation by global convolutional network”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, pgs. 1743-1751, 2017. DOI: [10.1109/CVPR.2017.189](https://doi.org/10.1109/CVPR.2017.189). arXiv: [1703.02719](https://arxiv.org/abs/1703.02719).
- [PEREIRA JUNIOR e WANGENHEIM 2019] P. C. PEREIRA JUNIOR e Aldo von WANGENHEIM. Orthomosaic Dataset of RGB aerial Images for Crop Rows Detection, 2019. Disponível em: <https://lapix.ufsc.br/crop-rows-sugar-cane/>.
- [PEREZ-ORTIZ *et al.* 2016] María PEREZ-ORTIZ, José M. PEÑA, Pedro A. GUTIERREZ, Jorge TORRES-SÁNCHEZ, César HERVÁS-MARTÍNEZ e Francisca LÓPEZ-GRANADOS. “Selecting patterns and features for between- and within- crop-row weed mapping using UAV-imagery”. *Expert Syst. Appl.* 2016, vol. 47, pgs. 85–94. DOI: [10.1016/j.eswa.2015.10.043](https://doi.org/10.1016/j.eswa.2015.10.043). URL: [pdf](#).
- [PINGEL *et al.* 2013] Thomas J. PINGEL, Keith C. CLARKE e William A. MCBRIDE. “An Improved Simple Morphological Filter for the Terrain Classification of Airborne LIDAR Data”. *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 77, 31-30, 2013. DOI: [10.1016/j.isprsjprs.2012.12.002](https://doi.org/10.1016/j.isprsjprs.2012.12.002).
- [PINHEIRO e COLLOBERT 2014] Pedro H. O. PINHEIRO e Renan COLLOBERT. “Recurrent convolutional neural networks for scene labeling”. Em: *ICML*, 2014. arXiv: [1306.2795](https://arxiv.org/abs/1306.2795).
- [PIX4DMANUAL] *PIX4D User Manual*. URL: <https://support.pix4d.com>.
- [PIX4DMAPPER] *Software PIX4Dmapper Discovery – PIX4D*. URL: <https://pix4d.com>.
- [POHLEN *et al.* 2017] Tobias POHLEN, Alexander HERMANS, Markus MATHIAS e Bastian LEIBE. “Full-resolution residual networks for semantic segmentation in street scenes”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, pgs. 3309-3318, 2017. DOI: [10.1109/CVPR.2017.353](https://doi.org/10.1109/CVPR.2017.353). arXiv: [1611.08323](https://arxiv.org/abs/1611.08323).
- [QGIS] Open-Source Geospatial Foundation. 2023. Quantum GIS (QGIS). Open-Source Geographic Information System (GIS). Version 3.28.2. Disponível em: <http://qgis.org/>.
- [RAWAT e WANG 2017] Waseem RAWAT e Zenghui WANG. “Deep convolutional neural networks for image classification: a comprehensive review”. Em: *Neural Computation*, vol. 29, n. 9, pgs. 2352-2449, 2017. DOI: [10.1162/NECO\\_a\\_00990](https://doi.org/10.1162/NECO_a_00990).
- [RENOVABIO] RenovaBio. Disponível em: <https://www.gov.br/mme/pt-br/assuntos/secretarias/petroleo-gas-natural-e-biocombustiveis/renovabio-1>. Último acesso em 15/12/2023.
- [RONNEBERGER *et al.* 2015] Olaf RONNEBERGER, Philipp FISCHER e Thomas BROX. “U-net: convolutional networks for biomedical image segmentation”. Em: *Proceedings*



of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI). Lecture Notes in Computer Science, vol. 9351, pgs. 234-241, 2015. DOI: [10.1007/978-3-319-24574-4\\_28](https://doi.org/10.1007/978-3-319-24574-4_28).

- [ROSENBLATT 1958] Frank ROSENBLATT. “The perceptron: a probabilistic model for information storage and organization in the brain”. Em: *Psychological review*, vol. 65, n. 6, pg. 386, 1958. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519).
- [ROSENBLATT 1961] Frank ROSENBLATT. Principles of neurodynamics, perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961. URL: [pdf](#).
- [ROUSE *et al.* 1974] J. W. ROUSE, R. H. HAAS, J. A. SCHELL e D. W. DEERING. “Monitoring vegetation systems in the great plains with ERTS”. *3rd Earth Resource Technology Satellite (ERTS)*, vol. 1, pgs. 48-62, 1974. URL: [pdf](#).
- [RUMELHART *et al.* 1986] David E. RUMELHART, Geoffrey E. HINTON e Ronald J. WILLIAMS. “Learning representations by back-propagating errors”. Em: *Nature*, vol. 323, n. 6088, pgs. 533-536, 1986. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [RUMELHART *et al.* 1986a] David E. RUMELHART, Geoffrey E. HINTON e Ronald J. WILLIAMS (1986). *Learning internal representations by error propagation*. Chapter 8 in *Parallel Distributed Processing: Foundations*. vol. 1, MIT Press, Cambridge, MA, pgs. 318–362. URL: [pdf](#).
- [RUPNIK *et al.* 2017] Ewelina RUPNIK, Mehdi DAAKI, Marc Pierrot DESEILLIGNY. MicMac—a free, open-source solution for photogrammetry. *Open geospatial data, software and standards*, vol. 2, n. 1, pgs. 1-9, 2017.
- [RUSSAKOVSKY *et al.* 2014] Olga RUSSAKOVSKY, Jia DENG, Hao SU, Jonathan KRAUSE, Sanjeev SATHEESH, Sean MA, Zhiheng HUANG, Andrej KARPATHY, Aditya KHOSLA, Michael BERNSTEIN, Alexander C. BERG e Li FEI-FEI. “ImageNet large scale visual recognition challenge (ILSVRC-2014)”, 2014. arXiv: [1409.0575](https://arxiv.org/abs/1409.0575). URL: <http://www.image-net.org/challenges/LSVRC/2014/>.
- [RUSSAKOVSKY *et al.* 2015] Olga RUSSAKOVSKY *et al.* “ImageNet large scale visual recognition challenge (ILSVRC-2015)”. Em: *International Journal of Computer Vision (IJCV)*, vol. 115, n. 3, pgs. 211-252, 2015. DOI: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y).
- [SA *et al.* 2017] Inkyu SA, Zetao CHEN, Marija POPOVIC, Raghav KHANNA, Frank LIEBISCH, Juan NIETO e Roland SIEGWART. “weedNet: dense semantic weed classification using multispectral images and MAV for smart farming”. *IEEE Robotics and Automation Letters*, vol. 3, n. 1, pgs. 588-595, 2017. URL: <https://github.com/inkyusa/weedNet>. arXiv: [1709.03329](https://arxiv.org/abs/1709.03329).
- [SA *et al.* 2018] Inkyu SA, Marija POPOVIC, Raghav KHANNA, Zetao CHEN, Philipp LOTTES, Frank LIEBISCH, Juan NIETO, Cyrill STACHNISS, Achim WALTER e Roland SIEGWART. “WeedMap: a large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming”. *Remote Sensing*, vol. 10, n. 9, pgs. 1423, 2018. DOI: [10.3390/rs10091423](https://doi.org/10.3390/rs10091423).
- [SANDINO *et al.* 2018] Juan SANDINO, Felipe Gonzalez, Kerrie Mengersen e Kevin J. Gaston. “UAVs and Machine Learning Revolutionising Invasive Grass and Vegetation Surveys in Remote Arid Lands”. *Sensors* 2018, 18, 605. URL: [pdf](#).
- [SANDLER *et al.* 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov e Liang-Chieh Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks”. *CVPR*, 2018. arXiv: [1801.04381](https://arxiv.org/abs/1801.04381).
- [SEIF 2019] George Seif. Semantic Segmentation Suite in TensorFlow. URL: <https://github.com/GeorgeSeif/Semantic-Segmentation-Suite/tree/master/models>.

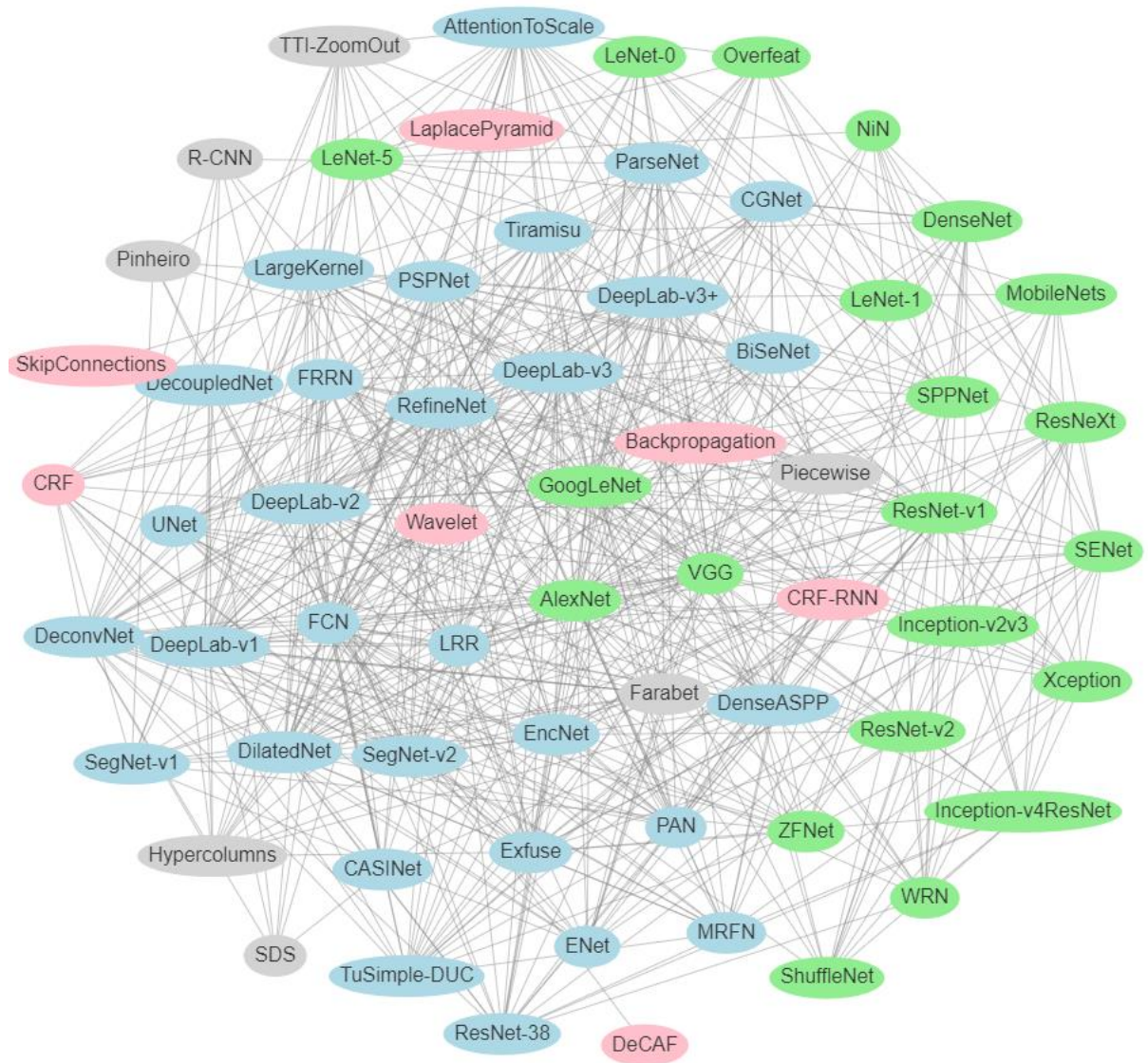


- [SERMANET *et al.* 2013] Pierre SERMANET, David EIGEN, Xiang ZHANG, Michael MATHIEU, Rob FERGUS e Yann LECUN. “OverFeat: integrated recognition, localization and detection using convolutional networks”. Em: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. arXiv: [1312.6229](https://arxiv.org/abs/1312.6229).
- [SHELHAMER *et al.* 2017] Evan SHELHAMER, Jonathan LONG e Trevor DARRELL. “Fully convolutional networks for semantic segmentation”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, n. 4, pgs. 640-651, 2017. DOI: [10.1109/TPAMI.2016.2572683](https://doi.org/10.1109/TPAMI.2016.2572683). arXiv: [1605.06211](https://arxiv.org/abs/1605.06211).
- [SHI *et al.* 2016] Yeyin SHI, J. Alex THOMASSON, Seth C. MURRAY, N. Ace PUGH, William L. ROONEY, Sanaz SHAFIAN, *et al.* “Unmanned aerial vehicles for high-throughput phenotyping and agronomic research”, *PLoS ONE*, vol. 11, n. 7, e0159781, 2016. DOI: [10.1371/journal.pone.0159781](https://doi.org/10.1371/journal.pone.0159781).
- [SHIRATSUCHI *et al.* 2014] Luciano S. SHIRATSUCHI, Ziany N. BRANDÃO, Luiz E. VICENTE, Daniel C. VICTORIA, Jorge R. DUCATI, Ronaldo P. de OLIVEIRA e Marina F. VILELA. *Sensoriamento remoto: conceitos básicos e aplicações na agricultura de precisão*. Em: A. C. C. BERNARDI, J. M. NAIME, A. V. RESENDE, L. H. BASSOI, R. Y. INAMASU (Ed.). *Agricultura de precisão: resultados de um novo olhar*. Brasília, DF: Embrapa, 2014. pgs. 58-73.
- [SILBERMAN *et al.* 2012] Nathan SILBERMAN, Derek HOIEM, Pushmeet KOHLI e Rob FERGUS. “Indoor segmentation and support inference from rgbd images”. Em: *ECCV*, pgs. 746-760, 2012. DOI: [10.1007/978-3-642-33715-4\\_54](https://doi.org/10.1007/978-3-642-33715-4_54). URL: [pdf](#).
- [SIMONYAN e ZISSERMAN 2015] Karen SIMONYAN e Andrew ZISSERMAN. “Very deep convolutional networks for large-scale image recognition”. Em: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. por Yoshua Bengio e Yann LeCun. 2015. arXiv: [1409.1556](https://arxiv.org/abs/1409.1556).
- [SMRF] smrf: A Simple Morphological Filter for Ground Identification of LIDAR Data. URL: <http://tpingel.org/code/smrf/smrf.html>
- [SONG *et al.* 2015] Shuran SONG, Samuel P. LICHTENBERG e Jianxiong XIAO. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. Em: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pgs. 567-576, 2015. DOI: [10.1109/CVPR.2015.7298655](https://doi.org/10.1109/CVPR.2015.7298655). URL: [pdf](#).
- [SRIVASTAVA *et al.* 2014] Nitish SRIVASTAVA, Geoffrey HINTON, Alex KRIZHEVSKY, Ilya SUTSKEVER e Ruslan SALAKHUTDINOV. “Dropout: a simple way to prevent neural networks from overfitting”. Em: *The Journal of Machine Learning Research*, vol. 15, n. 1, pgs. 1929-1958, 2014. URL: [pdf](#).
- [STRECHA *et al.* 2018] Christoph STRECHA, Luc Van GOOL e Pascal FUA. “A generative model for true orthorectification”. *Proceedings of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, 2008. URL: [pdf](#).
- [SUYKENS *et al.* 1999] Johan A. SUYKENS, Joos VANDEWALLE. “Least squares support vector machine classifiers”. *Neural processing letters*, Springer, vol. 9, n. 3, pgs. 293–300, 1999. URL: [pdf](#).
- [SUZUKI *et al.* 2010] Taro SUZUKI, Yoshiharu AMANO e Takumi HASHIZUME. “Vision based localization of a small UAV for generating a large mosaic image”, *Proceedings of SICE Annual Conference 2010*, pgs. 2960 – 2964.
- [SZEGEDY *et al.* 2015] Christian SZEGEDY, Wei LIU, Yangqing JIA, Pierre SERMANET, Scott REED, Dragomir ANGUELOV, Dumitru ERHAN, Vincent VANHOUCKE e Andrew RABINOVICH. “Going deeper with convolutions”. Em: *IEEE Conference on*

- Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, pgs. 1-9, 2015. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594). arXiv: [1409.4842](https://arxiv.org/abs/1409.4842).
- [SZEGEDY *et al.* 2016] Christian SZEGEDY, Vincent VANHOUCHE, Sergey IOFFE, Jon SHLENS e Zbigniew WOJNA. “Rethinking the inception architecture for computer vision”. Em: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pgs. 2818-2826. DOI: [10.1109/CVPR.2016.308](https://doi.org/10.1109/CVPR.2016.308). arXiv: [1512.00567](https://arxiv.org/abs/1512.00567).
- [SZEGEDY *et al.* 2017] Christian SZEGEDY, Sergey IOFFE, Vincent VANHOUCHE e Alexander A. Alemi. “Inception-v4, Inception-ResNet and the impact of residual connections on learning”. Em: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, California, 2017, pgs. 4278-4284. arXiv: [1602.07261](https://arxiv.org/abs/1602.07261).
- [TANG 2013] Yichuan TANG, “Deep learning using linear support vector machines”. Em: *Proceedings of the International Conference on Machine Learning (ICML) Workshops*, 2013. URL: [pdf](#).
- [TARG *et al.* 2016] Sasha TARG, Diogo ALMEIDA e Kevin LYMAN. “ResNet in ResNet: generalizing residual architectures”, 2016. arXiv: [1603.08029](https://arxiv.org/abs/1603.08029).
- [TENSORBOARD] TensorBoard: kit de ferramentas de visualização do TensorFlow. URL: <https://www.tensorflow.org/tensorboard/>
- [TOFFANIN 2023] Piero Toffanin. *OpenDroneMap: The Missing Guide* (2nd ed.). 2023.
- [TORRES-SÁNCHEZ *et al.* 2013] Jorge TORRES-SÁNCHEZ, Francisca LÓPEZ-GRANADOS, Ana I. de CASTRO e José M PEÑA-BARRAGÁN. “Configuration and specifications of an unmanned aerial vehicle (UAV) for early site specific weed management”. *PLoS ONE*, 8.3, e58210, 2013. DOI: [10.1371/journal.pone.0058210](https://doi.org/10.1371/journal.pone.0058210).
- [TRIGGS *et al.* 2000] Bill TRIGGS, Philip F. MCLAUCHLAN, Richard I. HARTLEY e Andrew W. FITZGIBBON. “Bundle adjustment – a modern synthesis”. In: *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms* Corfu, Greece, September 21–22, 1999 Proceedings. Springer Berlin Heidelberg, 2000. pgs. 298-372. URL: [pdf](#).
- [XIE *et al.* 2017] Saining XIE, Ross GIRSHICK, Piotr DOLLAR, Zhuowen TU e Kaiming HE. “Aggregated residual transformations for deep neural networks”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pgs. 5987-5995, 2017. DOI: [10.1109/CVPR.2017.634](https://doi.org/10.1109/CVPR.2017.634). arXiv: [1611.05431](https://arxiv.org/abs/1611.05431).
- [YANG *et al.* 2018] Maoke YANG, Kun YU, Chi ZHANG, Zhiwei LI e Kuiyuan YANG. “DenseASPP for semantic segmentation in street scenes”. Em: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, pgs. 3684-3692, 2018. DOI: [10.1109/CVPR.2018.00388](https://doi.org/10.1109/CVPR.2018.00388). URL: [pdf](#).
- [YOSINSKI *et al.* 2014] Jason YOSINSKI, Jeff CLUNE, Yoshua BENGIO e Hod LIPSON. How transferable are features in deep neural networks? Em: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. pgs. 3320–3328. arXiv: [1411.1792](https://arxiv.org/abs/1411.1792).
- [YU e KOLTUN 2016] Fisher YU e Vladlen KOLTUN. “Multi-scale context aggregation by dilated convolutions”. Em: *Proceedings of the 4th International Conference on Learning Representations, ICLR, San Juan, Puerto Rico*, 2016. arXiv: [1511.07122](https://arxiv.org/abs/1511.07122).
- [YU *et al.* 2018] Changqian YU, Jingbo WANG, Chao PENG, Changxin GAO, Gang YU e Nong SANG. “BiSeNet: Bilateral segmentation network for real-time semantic segmentation”. Em: *Proceedings of the 15th European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science, vol. 11217, pgs. 334-349, 2018. DOI: [10.1007/978-3-030-01261-8\\_20](https://doi.org/10.1007/978-3-030-01261-8_20). arXiv: [1808.00897](https://arxiv.org/abs/1808.00897).

- [YUAN *et al.* 2020] Jianlong YUAN, Zelu DENG, Shu WANG e Zhenbo LUO. “Multi receptive field network for semantic segmentation”. Em: *Proc. Winter Conf. Appl. Comput. Vis.*, pgs 1883–1892. IEEE, 2020. arXiv: [2011.08577](https://arxiv.org/abs/2011.08577).
- [WANG *et al.* 2018] Panqu WANG, Pengfei CHEN, Ye YUAN, Ding LIU, Zehua HUANG, Xiaodi HOU e Garrison COTTRELL. “Understanding convolution for semantic segmentation”. Em: *IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA*, pgs. 1451-1460, 2018. DOI: [10.1109/WACV.2018.00163](https://doi.org/10.1109/WACV.2018.00163). arXiv: [1702.08502](https://arxiv.org/abs/1702.08502).
- [WANG e BOVIK 2009] Zhou WANG e Alan C. BOVIK. “Mean squared error: Love it or leave it? a new look at signal fidelity measures”. *IEEE signal processing magazine*, IEEE, vol. 26, n. 1, pgs. 98–117, 2009. URL: [pdf](#).
- [WESTOBY *et al.* 2012] M. J. WESTOBY, J. BRASINGTON, N. F. GLASSER, M. J. HAMBREY e J. M. REYNOLDS. “Structure-from-Motion’ photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, vol. 179, pgs. 300–314, 2012. DOI: [10.1016/j.geomorph.2012.08.021](https://doi.org/10.1016/j.geomorph.2012.08.021).
- [WOLF *et al.* 2014] Paul R. WOLF, Bon A. DEWITT e Benjamin E. WILKINSON, *Elements of photogrammetry with applications in GIS*, 4 ed., McGraw Hill, 2014.
- [WU *et al.* 2019] Zifeng WU, Chunchua SHEN e Anton van den HENGEL. “Wider or deeper: revisiting the ResNet model for visual recognition”. Em: *Pattern Recognition*, vol. 90, pgs. 119-133, 2019. arXiv: [1611.10080](https://arxiv.org/abs/1611.10080).
- [ZAGORUYKO e KOMODAKIS 2016] Sergey ZAGORUYKO e Nikos KOMODAKIS. “Wide residual networks”. Em: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK*, 2016. arXiv: [1605.07146v3](https://arxiv.org/abs/1605.07146v3).
- [ZEILER e FERGUS 2014] Matthew D. ZEILER e Rob FERGUS. “Visualizing and understanding convolutional networks”. Em: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014, pgs. 818-833. DOI: [10.1007/978-3-319-10590-1\\_53](https://doi.org/10.1007/978-3-319-10590-1_53).
- [ZHANG *et al.* 2017] Xiangyu ZHANG, Xinyu ZHOU, Mengxiao LIN e Jian SUN. “Shufflenet: An extremely efficient convolutional neural network for mobile devices”. CoRR, abs/1707.01083, 2017. arXiv: [1707.01083](https://arxiv.org/abs/1707.01083).
- [ZHANG *et al.* 2018b] Zhenli ZHANG, Xiangyu ZHANG, Chao PENG, Dazhi CHENG, Jian SUN. “Exfuse: enhancing feature fusion for semantic segmentation”. Em: *ECCV*, 2018. arXiv: [1804.03821](https://arxiv.org/abs/1804.03821).
- [ZHANG *et al.* 2020] Aston ZHANG, Zachary C. LIPTON, Mu LI e Alexander J. SMOLA. *Dive into deep learning*. [S.l.]: Editora, 2020. URL: <https://d2l.ai/index.html>.
- [ZHANG e SABUNCU 2018] Zhilu ZHANG e Mert R. SABUNCU. “Generalized cross entropy loss for training deep neural networks with noisy labels”. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2018. pgs. 8778–8788. arXiv: [1805.07836](https://arxiv.org/abs/1805.07836).
- [ZHAO *et al.* 2017] Hengshuang ZHAO, Jianping SHI, Xiaojuan QI, Xiaogang WANG e Jiaya JIA. “Pyramid scene parsing network”. Em: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI*, pgs. 6230-6239, 2017. DOI: [10.1109/CVPR.2017.660](https://doi.org/10.1109/CVPR.2017.660). arXiv: [1612.01105](https://arxiv.org/abs/1612.01105).
- [ZHAO *et al.* 2018] Hengshuang ZHAO, Xiaojuan QI, Xiaoying SHEN, Jianping SHI e Jiaya JIA. “ICNet for real-time semantic segmentation on high-resolution images”. Em: *Computer Vision – Proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, Germany*, 2018. Lecture Notes in Computer Science, vol. 11207, pgs. 418-434, 2018. DOI: [10.1007/978-3-030-01219-9\\_25](https://doi.org/10.1007/978-3-030-01219-9_25).
- [ZHOU *et al.* 2014] B. ZHOU, H. ZHAO, X. PUIG, S. FIDLER, A. BARRIUSO e A. TORRALBA. “Semantic understanding of scenes through ADE20K dataset”. arXiv: [1608.05442](https://arxiv.org/abs/1608.05442), 2016.





“You can’t connect the dots looking forward; you can only connect them looking backwards. So, you have to trust that the dots will somehow connect in your future.”

Steve Jobs