An Analysis of Sample Synthesis for Deep Learning based Object Detection

Leonardo Blanger

Dissertation presented
to the
Institute of Mathematics and Statistics
of the
University of São Paulo
towards
obtaining the title
of
Master of Sciences

Computer Science Program

Supervisor: Prof. Nina S. T. Hirata

Throughout the development of this work, the author was funded by the
National Council for Scientific and Technological Development (CNPq)
under the grant nº 131802/2018-6,
and by the São Paulo Research Foundation (FAPESP)
under the grants nº 2018/00390-7 and 2019/17312-1.

São Paulo, November 2020

An Analysis of Sample Synthesis for Deep Learning based Object Detection

This version of the dissertation contains the corrections and updates suggested by the Evaluation Committee during the defense of the original version, which happened on October 16, 2020. A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

Evaluation Committee:

- Prof. Dr. Nina Sumiko Tomita Hirata (supervisor) - IME-USP
- Prof. Dr. David Menotti Gomes - UFPR
- Prof. Dr. Roberto de Alencar Lotufo - UNICAMP

Special Thanks

First of all, to my family, for all the support received during this period.

Next, I am forever grateful to my supervisor, Nina, who agreed to help me during these past three years, and to prof. Xiaoyi Jiang, who has greatly contributed to the ideas presented here.

Finally, this work would not be possible without the help from the National Council for Scientific and Technological Development (CNPq) and the São Paulo Research Foundation (FAPESP).

Abstract

BLANGER, L. **An Analysis of Sample Synthesis for Deep Learning based Object Detection**. 2020. Dissertation (Master) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2020.

This work investigates the use of artificially synthesized images as an attempt to reduce the dependency of modern Deep Learning based Object Detection techniques on expensive supervision. In particular, we propose using a big number of synthesized detection samples to pretrain Object Detection architectures before finetuning them on real detection data. As the major contribution of this project, we experimentally demonstrate how this pretraining works as a powerful initialization strategy, allowing the models to achieve competitive results using only a fraction of the original real labeled data. Additionally, in order to synthesize these samples, we propose a synthesis pipeline capable of generating an infinite stream of artificial images paired with bounding box annotations. We demonstrate how it is possible to design such a working synthesis pipeline just using already existing GAN techniques. Moreover, all stages in our synthesis pipeline can be fully trained using only classification images. Therefore, we managed to take advantage of bigger and cheaper classification datasets in order to improve results on the harder and more supervision hungry Object Detection problem. We demonstrate the effectiveness of this pretraining initialization strategy combined with the proposed synthesis pipeline, by performing detection using four real world objects: QR Codes, Faces, Birds and Cars.

Keywords: Object Detection, Sample Synthesis, Generative Models, Deep Learning.

Resumo

BLANGER L. **Uma Análise de Síntese de Exemplos para Detecção de Objetos baseada em *Deep Learning***. 2020. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, 2020.

Este trabalho investiga o uso de imagens sintetizadas como uma forma de reduzir a dependência de técnicas modernas de Detecção de Objetos, baseadas em *Deep Learning*, por formas caras de supervisão. Em particular, este trabalho propõe utilizar grandes quantidades de amostras de detecção sintetizadas para pré-treinar arquiteturas de Detecção de Objetos antes de ajustar estas arquiteturas usando dados reais. Como principal contribuição deste projeto, demonstramos experimentalmente como este pré-treinamento serve como uma poderosa estratégia de inicialização, permitindo que modelos atinjam resultados competitivos usando apenas uma fração dos dados rotulados reais. Além disso, para poder sintetizar estas amostras, propomos um pipeline de síntese capaz de gerar uma sequência infinita de imagens artificias associadas a anotações no formato de *bounding boxes*. Demonstramos como é possível projetar este pipeline de síntese usando apenas técnicas já existentes baseadas em GANs. Além disso, todos os estágios do nosso pipeline de síntese podem ser completamente treinados usando apenas imagens rotuladas para classificação. Desta forma, fomos capazes de tirar proveito de datasets maiores e mais baratos de rotular, para melhorar os resultados em Detecção de Objetos, um problema mais difícil e para o qual produzir dados rotulados diretamente é mais custoso. Demonstramos a eficácia desta estratégia de inicialização via pré-treinamento, em combinação com nosso pipeline de síntese, através de experimentos evolvendo detecção de quatro objetos reais: Códigos QR, Faces, Pássaros, e Carros.

Palavras Chave: Detecção de Objetos, Síntese de Amostras, Modelos Gerativos, Aprendizado Profundo, *Deep Learning*.

Contents

List of Abbreviations

CG	Computer Graphics
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CUB	Caltech-UCSD Birds
DM-GAN	Dynamic Memory GAN
DPM	Deformable Part-based Model
FCN	Fully Convolutional Network
FDDB	Face Detection Data Set and Benchmark
FFHQ	Flickr-Faces-HQ
GAN	Generative Adversarial Network
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Competition
IOU	Intersection over Union
LFW	Labeled Faces in the Wild
LR-GAN	Layered Recursive GAN
mAP	mean Average Precision
MS-COCO	Microsoft – Common Objects in Context
Pascal VOC	Pascal Visual Object Classes
QR Code	Quick Response Code
RCNN	Regions with CNN Features
ReDO	Unsupervised Object Segmentation by Redrawing
RPN	Region Proposal Network
SIFT	Scale Invariant Feature Transform
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine
YOLO	You Only Look Once

List of Figures

List of Tables

Chapter 1

Introduction

Since the recent widespread adoption of CNN based architectures for Computer Vision problems (Girshick *et al.*, 2014; Sermanet *et al.*, 2013), the performance of Object Detection techniques has been consistently improving. This is evidenced by the state-of-the-art results achieved by recent Deep Learning based detection architectures (Jiao *et al.*, 2019; Zhao *et al.*, 2019) on standard detection benchmarks such as Pascal VOC (Everingham *et al.*, 2015) and MS-COCO (Lin *et al.*, 2014). However, these Deep Learning architectures are well known for requiring huge amounts of labeled data in order to achieve their best performance and avoid the issue of overfitting. Therefore, despite their recent successes on these big standardized datasets, it remains difficult to deploy recent detection models on domains where labeled data is scarce.

Acquiring labeled data for Object Detection is difficult. In order to label images for detection, one needs to identify not only object categories, as is the case for classification, but also rectangular, axis-aligned bounding boxes (Everingham *et al.*, 2015), for each object in the image. The number of objects in a single image is variable and they may appear in varying sizes and shapes, potentially interacting or occluding each other in complex ways. All this variability makes the labeling effort harder, more time consuming, and more prone to mistakes and the influence of human biases than in the classification case.

In fact, for the related Image Classification problem, acquiring labeled data is much easier and faster, as evidenced by public datasets for classification (Deng *et al.*, 2009; Krizhevsky *et al.*, 2009), which tend to be orders of magnitude bigger than their detection counterparts (Everingham *et al.*, 2015; Lin *et al.*, 2014). Table 1.1 presents the number of samples for some popular Classification and Detection datasets.

Classification		**Detection**	
CIFAR-{10,100} (2009)	ImageNet (2010)	Pascal VOC 2012	MS-COCO 2017
60.000	14.197.122	11.530	∼123k

Table 1.1: *Number of images in some popular Classification and Detection datasets: CIFAR-{10,100} (Krizhevsky et al., 2009), ImageNet (Deng et al., 2009), Pascal VOC 2012 (Everingham et al., 2012), and MS-COCO (Lin et al., 2014).*

A core motivation of this project is to investigate whether it is possible to transfer the knowledge acquired from data coming from other tasks that require more high level and less costly forms of supervision, such as Image Classification, into the Object Detection task, in such a way to alleviate the need for the more expensive detection supervision. Assuming we somehow have access to (usually a large quantity) of classification level images from the given object classes to be detected, a central concern in this work is finding a way to use these images to automatically compose detection level images paired with bounding box labels, and then using these artificial detection images as training samples for traditional detection models.

Approaches to expand image datasets with artificial samples are not new. In the Deep Learning and Computer Vision communities, it has become common practice to apply Data Synthesis and Augmentation techniques in order to obtain additional training samples (Krizhevsky *et al.*, 2012; LeCun *et al.*, 1998; Liu *et al.*, 2016; Long *et al.*, 2015; Ronneberger *et al.*, 2015). Moreover, there is a line of works that use Computer Graphics techniques to synthesize artificial samples (Alhaija *et al.*, 2017; Gupta *et al.*, 2016; Hinterstoisser *et al.*, 2018; Mitash *et al.*, 2017; Movshovitz-Attias *et al.*, 2016; Peng *et al.*, 2015; Su *et al.*, 2015; Sun and Saenko, 2014; Varol *et al.*, 2017).

For Image Classification tasks, in recent years there have been attempts at using Generative Image Models to synthesize additional training samples (Antoniou *et al.*, 2017; Mariani *et al.*, 2018; Wang and Perez, 2017; Yamaguchi *et al.*, 2019; Zhang *et al.*, 2019), and this constitutes a promising research direction, especially in situations where labeled data is costly to acquire, like medical image applications.

Generative Adversarial Networks (GANs) (Goodfellow *et al.*, 2014) are a natural approach to perform image synthesis. In the past few years, GAN variations achieved impressive results on image generation and contributed to the recent surge of attention to Generative Image Models. Some recent iterations of these techniques are even capable of generating images close to indistinguishable from real ones to distracted human eyes (Brock *et al.*, 2018; Karras *et al.*, 2019a, 2020). Figure 1.1 shows some impressive examples of faces generated by the StyleGAN architecture (Karras *et al.*, 2019a). Additionally, the idea of taking advantage of the knowledge encoded by Generative Models about the data distribution in order to improve downstream tasks is an important Machine Learning goal.

The effectiveness of Generative Models for Image Classification Sample Synthesis suggests they might be a good choice for Object Detection as well. However, the few works that attempt to apply Generative Models to synthesize samples beyond Image Classification require heavy forms of annotation for training, like segmentation masks (Bailo *et al.*, 2019; Bowles *et al.*, 2018; Milz *et al.*, 2018), paired images from different domains (Sandfort *et al.*, 2019), or manually designed simulators (Shrivastava *et al.*, 2017). These approaches do not really help reduce the need for expensive supervision.

In this perspective, in this work we investigate how to use GAN based techniques to perform Object Detection Sample Synthesis. As we have this general motivation of taking

Figure 1.1: *Images Generated by a StyleGAN architecture trained on the FFHQ dataset (Karras et al., 2019a).*

advantage of the knowledge encoded on cheaply labeled classification datasets, while also reducing the need for expensive supervision like bounding boxes, we focused our investigation on how to design a synthesis pipeline in which all stages can be trained with classification level data only.

Additionally, once we have artificial detection samples, some questions arise. First, what is the best way to use these synthesized samples during the training of traditional detection models? Second, as we attempt to use artificially synthesized samples together with real images acquired from the real world, a natural question is how realistic these synthesized samples need to be in order to be useful. Or in other words, how much influence does sample quality have on the final detection results?

1.1 Goals and Contributions

The three main questions we seek to answer with this project are **(1)** Can we use Sample Synthesis to achieve more data efficient Object Detection? **(2)** How to design a synthesis pipeline that does not depend on bounding box labels or other forms of expensive supervision? and **(3)** What is the best way to incorporate these synthesized samples into the training of modern detection architectures?

In this work, we demonstrate how it is possible to combine already existing GAN inspired techniques into a simple synthesis pipeline capable of generating an infinite stream of artificial detection samples paired with bounding box labels. We take advantage of recent advances in Unsupervised Segmentation (Chen *et al.*, 2019) in order to avoid the need for expensive supervision. Therefore, all stages of our pipeline can be trained using only classification

level data. We also demonstrate how pretraining Object Detection models using this infinite stream of artificial samples works as a powerful initialization strategy, allowing comparable results using only a fraction of the original real labeled detection data.

In summary, the main contributions of this project are the following:

1. We show how Object Detection Sample Synthesis is a viable option to reduce the need for real labeled detection data, using only classification level supervision, therefore allowing us to take advantage of the knowledge encoded in large amounts of the cheaper classification data in order to improve data efficiency on the harder and more supervision hungry Object Detection problem.
2. We demonstrate how it is possible to achieve such a working synthesis pipeline using a simple combination of already existing GAN based techniques, and how this simple pipeline is capable of generating an infinite stream of synthesized detection images paired with bounding box labels.
3. We propose to use these synthesized samples as a pretraining initialization strategy, and present experimental evaluations to demonstrate the benefits of this method.

1.2 Dissertation Structure

The remainder of this text is structured as follows. Chapter 2 reviews the techniques on which this project is based, and establishes the common terminology that will be used throughout the text. Chapter 3 explains our synthesis pipeline in detail, together with a discussion of several other ideas we either considered or attempted, but did not make it to the final method, and could potentially be useful for future research. In Chapter 4 we present a series of experiments aimed at demonstrating how Sample Synthesis can help reduce the need for detection supervision, as well as demonstrating that our synthesis pipeline is a viable way to perform such synthesis. We also present a set of ablation experiments that aim at better understanding the influence of several design choices. To conclude, we summarize our findings in Chapter 5.

For additional information, in Appendix A we describe the main metric used for Object Detection evaluation. In Appendix B we describe in more detail the architectures used for the experiments, and in Appendix C we present all the main results in table format.

Additionally, in Appendix D we include a paper titled "An Evaluation of Deep Learning Techniques for QR Code Detection", published at ICIP 2019 (Blanger and Hirata, 2019). This paper, as well as an image dataset for QR Code detection [1] (detailed and used in Section 4.1) are contributions generated during preliminary studies towards understanding the modern Object Detection literature and Deep Learning programming tools.

[1] https://github.com/ImageU/QR_codes_dataset

Chapter 2

Related Work

In this chapter, we present an overview of the theoretical foundations on which this project is based, as well as related works that motivate the directions and choices taken in this project. Section 2.1 defines the basics of what we refer to as Object Detection, focusing especially on recent approaches that constitute the current state of the art. Section 2.2 explores one of the main limitations of these modern techniques: the requirement of large amounts of costly labeled data, and a series of existing attempts at mitigating this issue. In Section 2.3, we explore one of this mitigation attempts called "Data Synthesis", on which this project is based. Finally, Section 2.4 presents some recent trends in Machine Learning research that justify some of the choices we take for this project.

2.1 Object Detection

It is easier to describe what Object Detection is, by comparing it with other related Computer Vision tasks. In **Image Classification**, the goal is to assign the correct label from a predefined set to an input image. In the most classical formulation, images that are used for classification are either acquired with this intention or undergo some preprocessing to properly account for the scale of the object of interest. Ideally, in order to prevent ambiguous classification, the image contains a single object, which is guaranteed to belong to the set of predefined classes. Additionally, the object is also usually centralized and in large scale (it covers a significant region within the image frame), so as to minimize the interference of what is in the background.

Another Computer Vision task is **Object Localization**, which is similar to Image Classification in that each image still contains a single object from a predefined set of classes. However, for localization this object can be of varying scale, thus occupying a variable region within the image frame, and it can be located at varying positions. The goal in Object Localization is both to classify the object, while also locating it in the image, usually producing an output in the form of a rectangular, axis-aligned bounding box that tightly encloses the object. This usually means having to output four regression values parameterizing the

coordinates of this box.

Finally, the **Object Detection** problem can be seen as a generalization of Localization. In Detection, the goal is also to perform classification and bounding box regression for objects in images. However, contrary to Localization, here it is possible to have several objects of interest in the same image (possibly none at all). These objects can be of varying scales, positions and aspect ratios, and can interact, occlude and overlap each other in complex ways. The goal in Detection is to output a variable length list of classification and bounding box location pairs. A visual comparison between Classification, Localization and Detection is presented in Figure 2.1

We note that in some sources and fields, the terms "Localization" and "Detection" may be used interchangeably to refer to either one of the last two definitions or some other similar task. In this work however, we discriminate between them according to the definitions above. Whenever we refer to "Detection", we mean the whole task described above, and when we mention "Localization", we are referring to the specific task of a single bounding box regression, which is a component step of "Detection".

The Object Detection problem is considered to be significantly more difficult than Image Classification. This difficulty manifests in two different ways. First, the tasks a Detection model is required to perform are much more complex. In fact, Classification can be seen as a sub-task in the Detection problem, alongside the bounding box regression sub-task. Moreover, in the most traditional formulations, the Detection problem requires handling a variable number of highly variable objects of different scales, while Classification usually has to handle a well behaved single object at large scale. Second, due to this complexity, labeling data for training Detection models is considerably harder and more time consuming than for Classification, and this results in datasets for Detection problems being smaller. Therefore, despite the higher complexity of the problem, Detection faces a lower data availability than Classification. This situation will be expanded on Section 2.2

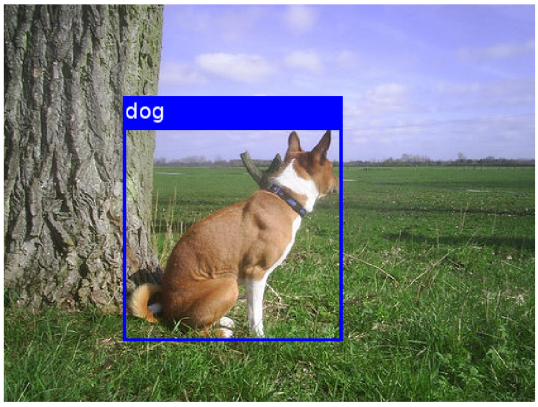
In this project, we assumed the same formulation of Object Detection as the one used in the detection track of the Pascal VOC competition (Everingham *et al.*, 2015). For all the evaluations presented in Chapter 4, we measured results using Mean Average Precision at 0.5 IOU, following the same methodology as done since the 2010 edition of the VOC competition, which is described in Everingham *et al.* (2010). For details about how to calculate Mean Average Precision in the context of Object Detection, please refer to Appendix A.

2.1.1 Traditional vs Deep Learning Approaches

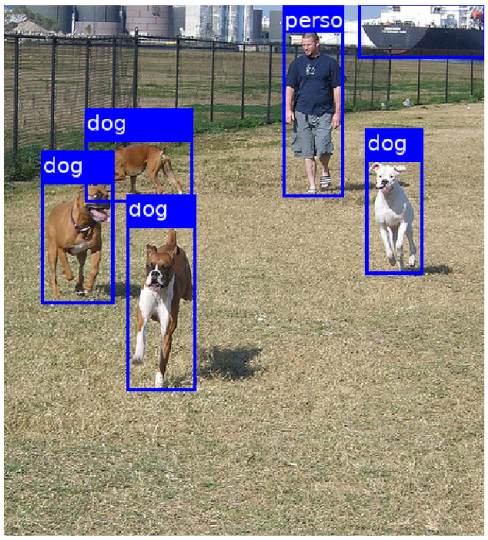
As mentioned previously, the Object Detection problem can be divided into two sub-tasks: identifying the image region on which a given object is located (sometimes also called localization), and identifying to which class from a predefined set the object belongs (classification). According to Zhao *et al.* (2019), traditional detection pipelines usually accomplish both of these tasks using a sequence of three main steps:

Class: DOG

(a)



(b)



(c)

Figure 2.1: *An illustration about the distinction between Image Classification (a), Localization (b), and Object Detection (c). The image is the input for the task, while everything in blue is the respective expected output. Images taken from the Pascal VOC 2012 detection dataset (Everingham et al., 2012)*

Selection of Informative Regions. Objects can occur at any image region and at any scale or aspect ratio. To handle this situation, it was common for traditional techniques to scan the image in a sliding window fashion at multiple scales, in order to extract candidate regions for detection. This approach has the downside of requiring the application of some form of classifier several times, which is not practical for more computationally expensive models.

Feature Extraction. It was common practice to use feature extractors manually designed for each particular domain. However, due to the high variety of appearances, lighting and background, it was hard to manually design a robust enough set of features for multiple object classes in generic contexts (Zhao *et al.*, 2019). Some popular attempts at generic feature extraction include SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005) and Haar features (Lienhart and Maydt, 2002)

Classification. In order to decide which candidate regions are associated with each object class, a general purpose classifier is usually applied to the previously extracted region features. For Object Detection tasks, Support Vector Machines (SVMs) (Cortes and Vapnik, 1995), AdaBoost (Freund and Schapire, 1997), and Deformable Part-based Models (DPMs) (Felzenszwalb *et al.*, 2009) were frequently used options for Object Detection (Zhao *et al.*, 2019).

Throughout the 2000's decade, several advances were achieved by employing techniques based on this sequence of steps. A representative example is the detection framework of Viola & Jones (Viola *et al.*, 2001), that uses a classifier cascade on top of Haar features (Lienhart and Maydt, 2002). Proposed in 2001, this framework achieved impressive results on Face detection, and has since been applied to several other problems as well, like QR Code detection (Belussi and Hirata, 2013).

At around 2010-2012, only small incremental improvements were being achieved, mostly by using ensembles and small tweaks of already popular techniques (Zhao *et al.*, 2019). At around this period, in 2012, Krizhevsky *et al.* (2012) won the Image Classification competition ILSVRC (Russakovsky *et al.*, 2015) using a Deep Convolutional Neural Network (CNN) (LeCun *et al.*, 1990) trained on the now famous Imagenet dataset (Deng *et al.*, 2009). Since then, deep neural networks have been applied to several other problems, in Computer Vision and other fields. This strategy of applying deep layered forms of pattern extraction using neural network techniques eventually came to be called **Deep Learning**.

One of the first representative works that adopted these modern Deep Learning based techniques for Object Detection is the Overfeat architecture (Sermanet *et al.*, 2013), an expansion of the winner architecture used by Krizhevsky *et al.* (2012), which was originally trained for classification. Overfeat appends a localization module to the CNN, which acts on top of the feature maps extracted from the last convolutional layer. This additional module is trained to perform regression of the four bounding box coordinates. With this, the original classification parts and the new localization module act together sharing the same set of

convolutional feature maps. In order to perform detection, Overfeat explores the fact that CNNs process the input image in a sort of implicit sliding window, which allows it to detect objects at several regions with a single pass over the image. However, in order to deal with objects of varying scale, they still had to apply this architecture several times using an image pyramid approach.

The techniques that followed can be roughly divided into two categories (Zhao *et al.*, 2019), according to whether or not they have a candidate region extraction step:

Two-Stage Techniques. Techniques in this category perform detection by first searching for candidate regions in the image, and then classifying/discarding these regions and, possibly, refining their bounding box coordinates. The most representative works from this category are the ones from the RCNN family (Girshick, 2015; Girshick *et al.*, 2014; Ren *et al.*, 2015). The initial RCNN (Girshick *et al.*, 2014) (*Regions with CNN features*) consisted of a CNN inserted into a traditional detection pipeline. The technique depended on some algorithm capable of extracting candidate regions (in the original work they used *Selective Search* (Uijlings *et al.*, 2013) to extract around 2k regions, but follow up works also used the *Edge Boxes* (Zitnick and Dollár, 2014) method). These regions were then all resized to the same predefined dimensions and passed through a CNN for feature extraction and classification.

The last iterations of two-stage techniques are all instances of the Faster-RCNN architecture (Ren *et al.*, 2015), which uses a fully neural *Region Proposal Network* (RPN) instead of classical algorithms to extract candidate regions. Additionally, the components that perform the final detection (classify the candidate regions and refine the box coordinates) share most of their layers with the RPN, therefore significantly reducing the computational cost exclusively dedicated to region proposal.

Historically, two-stage techniques have achieved superior quantitative results when compared to single-stage techniques, at the cost of being more computationally inefficient and less stable to train (Zhao *et al.*, 2019).

Single-Stage Techniques. Contrary to the two-stage approaches, techniques in this group do not extract and refine candidate regions, and instead produce detections with a single network pass over the input image. Overall, single-state techniques generate one or more dense detection grids, containing class scores (including for the negative class, or background) with bounding box offsets with respect to some predefined set of anchor or reference boxes.

A pioneering single-stage technique is YOLO (Redmon *et al.*, 2016) (*You Only Look Once*), that outputs a dense grid of detections as described above. Each cell of this detection grid can detect up to a predefined number B of objects, and is responsible for both classifying them and adjusting bounding box coordinates.

Another popular single-stage technique is SSD (Liu *et al.*, 2016) (*Single Shot Multibox Detector*), that produces several dense detection grids. Each grid is generated by a detection branch on top of a feature map extracted from some layer of a backbone CNN. SSD exploits

the fact that more shallow CNN layers extract simple, low level patterns on higher resolution grids where each cell covers a small image region, while deeper layers extract more abstract high level patterns, on smaller grids where each cell covers a bigger part of the image. This way, by using multiple detection branches on top of different feature depths, SSD can naturally detect objects of varying scales.

Historically, single-stage approaches have performed worse than two-stage ones in terms of detection quality, but they are known to be more computationally efficient and easier to train (Zhao *et al.*, 2019). More recently however, some techniques were proposed to reduce this quality gap. An important advance comes from the adoption of Focal Loss (Lin *et al.*, 2017) for training instead of the traditional Cross Entropy Loss. Focal Loss aims at reducing the effects of class imbalance, an issue that particularly affects single-stage techniques, since the detection grids usually contain an overwhelmingly higher number of negative background matches than positive ones.

For simplicity purposes, in this work we choose to focus only on single-stage techniques for our experiments.

2.1.2 Image Segmentation

Another Computer Vision problem related to Object Detection that is worth mentioning is that of Image Segmentation. In it, the goal is to classify individual pixels into a set of classes. The output of a Segmentation model is one or more masks that segment the image region that is covered by a given object instance or object class. Because of this, it requires an even lower level form of supervision for training than Object Detection, usually in the form of ground truth masks.

There are two main styles of Image Segmentation, which we describe next:

Semantic Segmentation. Here the goal is to assign a single category to each image pixel. It does not differentiate between different instances of the object class. For instance, if there are two cars on an image, all the pixels within any of the car's regions should be assigned to the "car" class, and there is no reference as to which car each pixel belongs.

Instance Segmentation. Here the model output is expected to incorporate the information about individual instances of the same object class. For instance, in the situation described previously, the instance segmentation output should contain a mask for each individual car, segmenting only that particular car's region.

Representative works that perform different forms of Image Segmentation using Deep Learning are the Unet (Ronneberger *et al.*, 2015), FCN (Long *et al.*, 2015), and Mask-RCNN (He *et al.*, 2017).

2.2 The Data Scarcity Problem

Recent Machine Learning techniques that are based on the Deep Learning paradigm allowed applications to take advantage of increasingly large amounts of data and computing resources, which in turn has allowed them to achieve results previously thought to be out of reach for machines. However, the applicability of these techniques on domains with little labeled data is still limited. Many of the impressive results achieved recently stem from very big models with huge parameter spaces, which in turn requires massive amounts of (usually labeled) data in order to avoid overfitting (Shorten and Khoshgoftaar, 2019).

This leads to a situation where we have impressive results on areas where there are big datasets and standardized benchmarks. In contrast, we observe more limited improvements for domains where data is difficult or expensive to acquire due to technical or legal requirements, where the effort required for labeling data is higher or demands specialized knowledge, with the most notable examples being medical applications.

This problem is also present, although at a smaller degree, for Computer Vision tasks in general. For the traditional Image Classification, image samples are relatively easy to obtain, and providing labels for these samples is also fast and trivial most of the time. As for Object Detection, however, labeling images is considerably costlier, as the labeler needs to identify not only object categories, but also rectangular, axis-aligned bounding boxes that tightly enclose the given objects. This must be done for several objects that interact in complex ways in each image (Everingham *et al.*, 2015), thus making the labeling effort harder, more time consuming and more prone to mistakes and human biases than in the classification case.

This situation becomes evident when considering existing public datasets for Image Classification like ImageNet (Deng *et al.*, 2009), which tend to be orders of magnitude bigger than their Detection counterparts, such as Pascal VOC (Everingham *et al.*, 2015) and MS-COCO (Lin *et al.*, 2014). This situation is highlighted in the Introduction chapter of this text.

In the remainder of this section, we briefly detail some popular approaches used to alleviate this problem of lacking labeled data. We focus only on approaches that are popular in the Computer Vision literature, particularly those that remain or became widely used since the introduction of Deep Learning.

Improved Architectures and Regularization

A common approach to take better advantage of small datasets and in turn reduce overfitting, is to artificially restrict the model capacity in a controlled way. In the Machine Learning literature, this form of restriction is known as Regularization.

A traditional technique to perform regularization on neural networks is known as "Weight Decay", in which an L1 or L2 norm penalty is added to the loss function to prevent the model parameters from growing too much. Notable regularization techniques post Deep Learning include forms of Dropout (Srivastava *et al.*, 2014), in which the activations of certain units

in the model are forcibly set to zero during training, with the objective of preventing co-adaptation between units and thus increasing individual unit's robustness. Another popular form of regularization post Deep Learning is Batch Normalization (Ioffe and Szegedy, 2015), in which the activation output of certain layers in a model is standardized across the batch dimension, and then rescaled and shifted according to a scale and bias parameters, with the goal of reducing a problematic optimization phenomenon known as "Covariate Shift".

Besides regularization, there have been attempts at designing better parameter efficient architectures, with the goals of increased memory efficiency and better generalization capabilities. Notable architectures are the MobileNet (Howard *et al.*, 2019, 2017; Sandler *et al.*, 2018) and ShuffleNet (Ma *et al.*, 2018; Zhang *et al.*, 2018) series, respectively.

Image Augmentation

In the face of little labeled data, approaches to expand existing datasets by artificially generating additional samples have played an important role in Computer Vision problems so far. Traditionally, these approaches fall under the class of Data Augmentation techniques (Shorten and Khoshgoftaar, 2019), and some forms of it are present since the early applications of convolutional layers (LeCun *et al.*, 1998).

The traditional applications of data augmentation on Computer Vision consist of simple, manually designed transformations that change the images in a "label-preserving" way, like random color or geometric manipulations, random cropping or erasing of patches, and the injection of different forms of noise (Shorten and Khoshgoftaar, 2019). Several important works in different areas of Computer Vision made use of some form of these simple augmentations, including both early (LeCun *et al.*, 1998) and modern (Krizhevsky *et al.*, 2012) applications of CNNs for classification, detection (Liu *et al.*, 2016), and segmentation (Long *et al.*, 2015; Ronneberger *et al.*, 2015).

More recently, however, some works attempted to learn data augmentation strategies directly from data. One possibility is learning sampling policies over sets of manually designed traditional operations. For instance, Cubuk *et al.* (2019), Geng *et al.* (2018), Minh *et al.* (2018), and Zoph *et al.* (2019) all used Reinforcement Learning algorithms in order to navigate the space of augmentation policies, while Ratner *et al.* (2017) used a GAN-like framework to adversarially train a generator responsible for sampling sequences of predefined transformation functions.

Another set of works attempted to learn how to modify existing samples from scratch, without requiring manually designed transformation sets. Jackson *et al.* (2018) used a Style Transfer algorithm (Ghiasi *et al.*, 2017) to generate additional images by changing the style of existing ones. The Neural Augmentation (Wang and Perez, 2017) and Smart Augmentation (Lemley *et al.*, 2017) techniques work by using networks responsible for taking one or more images as input and generating one or more combined images that are treated as augmented samples and used to train some downstream classifier.

Transfer Learning

Transfer Learning techniques are concerned with taking advantage of the knowledge acquired in some domain with high data availability, in a way that improves some different but related task where there is less data (Weiss *et al.*, 2016). For instance, in certain Computer Vision problems, it became common practice to reuse parts of CNN architectures previously trained for classification on ImageNet as feature extractors (or backbone CNNs) instead of initializing them at random. This is done in the R-CNN (Girshick *et al.*, 2014) and SSD (Liu *et al.*, 2016) architectures for Object Detection and in the FCN (Long *et al.*, 2015) for Semantic Segmentation. It turns out that there is some form of "universality" on the patterns learned by convolutional layers, so they can be trained with large datsets on some task like ImageNet classification and these patterns would then generalize to other vision problems.

2.3 Data Synthesis

In the previous sections, we described the Object Detection problem, and how the cost of labeling detection data slows its applicability. We also compiled some common approaches for tackling this issue. In this section, we describe in more detail another of these approaches, under which we are categorizing this project.

As established in the Introduction, in this work we set out to investigate if it is possible to improve either the final results of Object Detection models, or their dependence on real labeled data, by using additional synthesized samples. In this section, we describe what we refer to as "Data Synthesis", presenting some existing possibilities and their drawbacks, with a focus on Object Detection, while also establishing the constraints we would like to have on our Data Synthesis process.

First, it is important to establish the difference between Data Synthesis and Augmentation. In this project, we refer as augmentation to any technique that generates additional training samples by applying random, label preserving transformations, to already existing samples from the same problem domain. In contrast, we assume Data Synthesis to be a broader definition, where additional samples are also generated, but not necessarily as transformations of existing ones. We note that, however, some works use Augmentation to refer to any data generation procedure.

As the goal of "Data Synthesis" is to somehow generate (image) samples, a natural approach is to use Computer Graphics techniques to model the objects of interest. One could use these models to generate several object instances to compose synthesized samples. Several existing works demonstrate promising results with variations of these general idea (Alhaija *et al.*, 2017; Gupta *et al.*, 2016; Hinterstoisser *et al.*, 2018; Mitash *et al.*, 2017; Movshovitz-Attias *et al.*, 2016; Peng *et al.*, 2015; Su *et al.*, 2015; Sun and Saenko, 2014; Varol *et al.*, 2017). However, these CG based techniques all require access to 3D models of the object of interest in order to be applied.

Another natural approach to perform "Data Synthesis" is through Generative Image Models, as done by several existing works, mostly with GANs (Goodfellow *et al.*, 2014). Many works use generative models to synthesize samples to expand Image Classification datasets (Antoniou *et al.*, 2017; Mariani *et al.*, 2018; Wang and Perez, 2017; Yamaguchi *et al.*, 2019; Zhang *et al.*, 2019), and this constitutes a promising research direction, specially on domains like medical imaging (Frid-Adar *et al.*, 2018), or situations with underrepresented classes (Lim *et al.*, 2018; Pinetz *et al.*, 2019; Zhu *et al.*, 2018).

Some works tried to apply GAN based synthesis (in some cases data augmentation) on tasks beyond just classification, such as detection and segmentation on medical (Bailo *et al.*, 2019; Bowles *et al.*, 2018; Sandfort *et al.*, 2019) and aerial images (Milz *et al.*, 2018), as well as eye gaze and hand pose estimation (Shrivastava *et al.*, 2017). However, these approaches require costly forms of supervision in order to train the GANs, like segmentation masks (Bailo *et al.*, 2019; Bowles *et al.*, 2018; Milz *et al.*, 2018), paired images from different domains (Sandfort *et al.*, 2019), or manually designed simulators to guide the generation process (Shrivastava *et al.*, 2017). Therefore, for our purposes, these approaches can not be used to reduce the dependency on expensive real labeled data.

The main challenge of Sample Synthesis for Object Detection is the requirement to generate granular forms of supervision, like bounding boxes, along with the synthesized images. Current generative image models perform very well on single, centralized and full image objects (Brock *et al.*, 2018; Karras *et al.*, 2019a, 2020; Razavi *et al.*, 2019), but struggle to generate images with multiple objects of varying scales and geometric configurations interacting in complex scenes, while at the same time generating coherent bounding box labels. Existing attempts either are constrained to simplified artificial domains (Arandjelović and Zisserman, 2019), or require expensive forms of supervision like bounding boxes (Hinz *et al.*, 2019; Reed *et al.*, 2016b), segmentation masks (Reed *et al.*, 2016a; Turkoglu *et al.*, 2019; Wang *et al.*, 2018), or key points (Reed *et al.*, 2016a,b) for training.

A more feasible approach would be to start from a real background image, and then generate objects coherently in it. This is the approach taken in Hong *et al.* (2018) and Park *et al.* (2018). However, these methods still require images labeled with bounding boxes and segmentation masks for training, respectively.

This is also the approach we take in this project, with the exception that we restrict the synthesis process to depend only on classification level supervision. For that, we expand upon a recently proposed unsupervised segmentation technique (Chen *et al.*, 2019) to perform sample synthesis for object detection. Our method manages to synthesize bounding box annotated images, while using only classification images for training.

We present more details regarding our method, as well as related attempts at Data Synthesis for Object Detection, in Chapter 3.

2.4 Extracting Knowledge from Cheaper Supervision

An important characteristic of our method is the fact that it uses only classification level supervision for training. This characteristic is aligned with a broader Machine Learning trend of methods that take advantage of large quantities of data labeled with cheaper forms of supervision, as a way to reduce the need for expensive annotations. This trend manifests itself in several different ways in different domains, but in the context of Computer Vision, it usually assumes the form of Unsupervised/Semi-Supervised Pretraining.

Semi-Supervised Pretraining can be classified as an Unsupervised technique that operates using Supervised training methods. A common formulation consists of initializing a model by pretraining it to perform some form of supervised "pretext" task, for which the labels can be extracted directly from the images given the definition of this pretext task. This way, this pretext pretraining can be done completely using unlabeled data. The goal of this pretraining initialization is that, despite the results on the pretext task being irrelevant, the model is forced to learn useful feature extractors that generalize well to the final downstream problem (Weng, 2019).

For instance, Dosovitskiy *et al.* (2014) sampled a set of small image patches, and applied several distortions to each of these patches. The pretext task's goal was then to predict which of the original patches a given distorted patch came from. Gidaris *et al.* (2018) applied rotations (one of 0°, 90°, 180°, 270°) to unlabeled images, and pretrained a model to classify these rotation angles. Doersch *et al.* (2015) sampled two adjacent image patches and pretrained a model to predict the relative position between these patches (top-left, ..., bottom-right). Noroozi and Favaro (2016) also used image patches, but the model was pretrained to output the correct order of a given shuffled sequence of input patches. Zhang *et al.* (2016) pretrained a model to reconstruct colorful images based on grayscale inputs.

Similar ideas are also present in the context of Generative Image Modeling. For instance, Denoising Autoencoders (Vincent *et al.*, 2008) are trained to reconstruct images that were partially corrupted (for instance, with the addition of noise). Context Encoders (Pathak *et al.*, 2016) are trained to draw missing patches of images given their surrounding context. The Split-Brain Autoencoder (Zhang *et al.*, 2017) does something similar, with the model being trained to reconstruct missing image channels from existing ones.

As we can see, there are several different ways of exploiting cheaper supervision, with the only constant being having access to big amounts of data that is not necessarily labeled for the final task, but that is useful nonetheless if properly taken advantage of. For additional information about existing Semi-Supervised works, please refer to the survey conducted by Weng (2019).

Our work relates to this general trend in a few ways. First, we aim at taking advantage of a cheaper form of supervision (classification labels) for a more strongly supervised task (Object Detection). Despite still requiring some form of labels, labeled images for classification are significantly cheaper to acquire than for detection, and thus are available in bigger quantities.

Second, we also found it beneficial to use our synthesized samples to pretrain detection models before finetuning them using real labeled data (as shown in Chapter 4). Despite the fact that this pretraining is performed on the same task as the final problem (detection), the synthesized samples come from an infinite stream of automatically labeled samples (details in Chapter 3), in a way similar to automatically extracted labels from pretext tasks.

2.5 Final Remarks

In the previous Sections, we defined what we refer to as Object Detection (Section 2.1), and presented the two styles that are currently used by detection architectures, single stage and two stages. We discussed the data scarcity problem, and why labeling data for Object Detection is costly (Section 2.2). We also described a series of popular approaches currently used to tackle this issue. We then described one of these approaches, Data Synthesis, in more detail, as it is the major focus of this project (Section 2.3). We noted how several existing Data Synthesis methods are based on Generative Image Models.

We also noted that, for tasks beyond Image Classification, existing Data Synthesis approaches tend to require expensive forms of supervision, like bounding boxes or segmentation masks. This makes it hard to perform Data Synthesis on domains with already low availability of labeled data. Therefore, an approach capable of performing Object Detection Sample Synthesis using only cheap classification annotations would be aligned with the current Machine Learning trends of Unsupervised/Semi-supervised Pretraining (Section 2.4) of taking advantage of cheaper supervision for downstream harder problems.

In this perspective, this project investigates whether it is possible to perform Object Detection Sample Synthesis using only cheaper classification images. For this, we show how it is possible to design a synthesis pipeline using already existing GAN based components, in such a way that it uses only classification supervision. We show in Chapter 4 that pretraining deep detection models on these cheap synthesized samples works as a powerful initialization before training on real data, allowing comparable results with less labeled real samples. After pretraining, all the traditional detection techniques such as augmentation are still applicable. Details of our pipeline are presented in Chapter 3, while experiments measuring the feasibility of Object Detection Sample Synthesis are presented in Chapter 4.

Chapter 3

Our Proposal

We set out to investigate if there is a way to perform sample synthesis for Object Detection that does not require expensive supervision. In this chapter, we first list a sequence of directions we either experimentally attempted or at least considered theoretically (Section 3.1). These attempts are listed in an order as to make sense of the final decisions we took, starting from a more abstract and general approach, and gradually adding restrictions and simplifying the problem according to the limitations we found on current techniques. Next, we briefly review some techniques that have the potential to address these limitations (Section 3.2). We then describe our approach to this problem, as well as the assumptions and simplifications we had to make in order to achieve a working method (Section 3.3).

3.1 Attempts at Object Detection Sample Synthesis

3.1.1 Direct Generation

A natural starting point would be trying to somehow use modern generative image models for our purposes. This field achieved impressive progress in the past few years, since the start of the Deep Learning era. The most representative classes of techniques are currently Generative Adversarial Networks (Goodfellow *et al.*, 2014), Variational Auto Encoders (Kingma and Welling, 2013), and Auto Regressive Models (Van Oord *et al.*, 2016). More recently, the last iterations of these methods have even shown to be capable of generating images that are close to indistinguishable to real ones by human eyes (Karras *et al.*, 2019a,b).

Formulating the Object Detection Sample Synthesis problem directly as a generative problem simply means designing a generative architecture that outputs images containing the objects of interest alongside lists of bounding box labels for these objects. This is illustrated for a GAN generator in Figure 3.1.

Generative Image Models have already been applied for sample synthesis for image classification (Antoniou *et al.*, 2017; Frid-Adar *et al.*, 2018; Lim *et al.*, 2018; Mariani *et al.*, 2018; Pinetz *et al.*, 2019; Wang and Perez, 2017; Yamaguchi *et al.*, 2019; Zhang *et al.*, 2019;

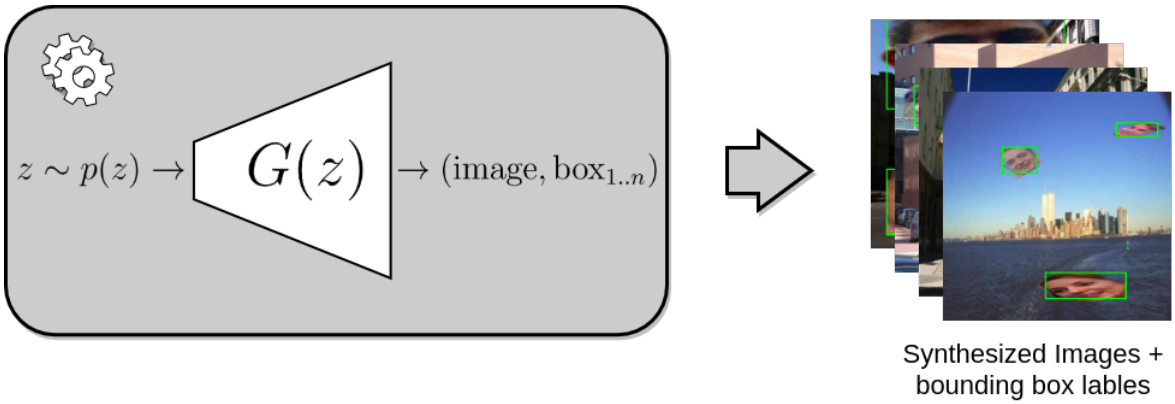


Figure 3.1: *Direct Generation of Object Detection samples using a GAN generator.*

Zhu *et al.*, 2018), so to do something similar for detection seems like a natural extension. However, the straightforward application of generative models for synthesizing detection samples stumbles on several issues. Most significantly, samples for classification are usually central crops of the object of interest, with a single object close to the center and covering a big portion of the image. Samples for detection however, are way more flexible, and can have several different objects that belong to different classes on top of a background scene. These objects can be of varying scales and aspect ratios and appear at apparently random positions and poses, while also interacting or occluding each other in complex ways. Current generative techniques are still trained exclusively with classification level data, and therefore, they can only produce classification images.

3.1.2 Controlling the Output

There have been some attempts to have control over some aspects of the generated image (beyond the simple class conditioning), including position and size. Bau *et al.* (2019) proposed a method called *GAN Dissection*, in which it is possible to manipulate the output of a pretrained GAN generator by either dropping or increasing the activation of some feature maps in the units corresponding to the manipulated region. With this, it is possible to "erase" objects or structures in the generated image, and to introduce these structures in some other image region. This approach is illustrated in Figure 3.2.

This approach of directly manipulating the generation process is promising for synthesizing detection samples in the long term, but there are several limitations that prevent it from being a viable option for our current purposes. First, the existing implementations are still restricted to manipulating simple structures, like trees, doors, buildings, etc. Second, in the current formulation, it is not clear how we could extract bounding box annotations from the generated images, which we need in order to produce synthesized labeled samples.

In summary, to the best of our knowledge, current generative image models are not advanced enough yet to perform either this "direct generation" nor anything that requires generating the full image at once.

Figure 3.2: *GAN Dissection.*
Image taken from Bau et al. (2019).

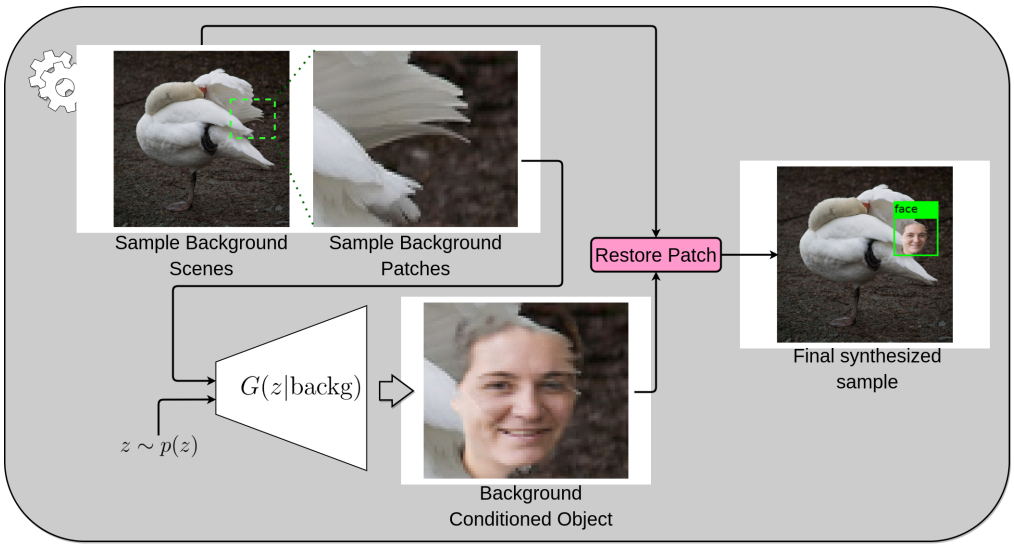


Figure 3.3: *Background Conditioned Sample Synthesis*

3.1.3 Generating Objects Conditioned on Backgrounds

Since generating full images with bounding box annotations is unlikely to be feasible with current technology, we have been led to seek a simpler formulation for synthesizing samples. In particular, instead of generating images labeled for detection from scratch, we considered starting from background scenes, sampling random patches from these scenes, and using some form of generative model to draw the objects of interest within these patches, conditioned on the patch's background content. The background scenes could either be real images or created by some other generative procedure. This is demonstrated on diagram in Figure 3.3.

One important requirement for this to work would be to ensure that the object generation step keeps the background context around the object coherent with the rest of scene. In our attempts however, we found it extremely hard to design a GAN that could achieve this result satisfactorily. By using existing Conditional GAN techniques, we found that the generator either failed to generate a minimally realistic object, or completely covered the existing background. We tried several architectural tweaks to allow the discriminator to notice when

(a)

(b)

(c)

Figure 3.4: *Five simple background textures (a), some samples generated by background conditioned GANs whose generators output a central patch region (b), and a central patch region residual (c), respectively. Besides the lack of realism, we can also notice a high degree of mode collapse, with "digits" on top of the same background looking very similar to each other.*

the generator was ignoring the background patch, but all required careful hyperparameter tuning and training was considerably unstable.

Figure 3.4 shows some instances produced by two different failed attempts at a very simple task. The goal here was simply to draw an MNIST digit on top of one of the given background textures in Figure 3.4a. In all cases, the samples were produced by feeding the background patch to a GAN generator as an additional input. The samples in Figure 3.4b were produced by a generator trained to output the central patch region, while the ones in Figure 3.4c were produced by a generator trained to output a "patch residual" that was added to the original central patch region. In both situations, the generator only modifies a central patch region so that the discriminator can have access to at least some of the original background. Without this, we observed that the generator mode-collapses into generating all digits on top of a single texture and simply overwrites the background.

In the toy problems where this approach (partially) worked, we stumbled upon two additional practical problems. The first is that we needed to generate objects in different scales,

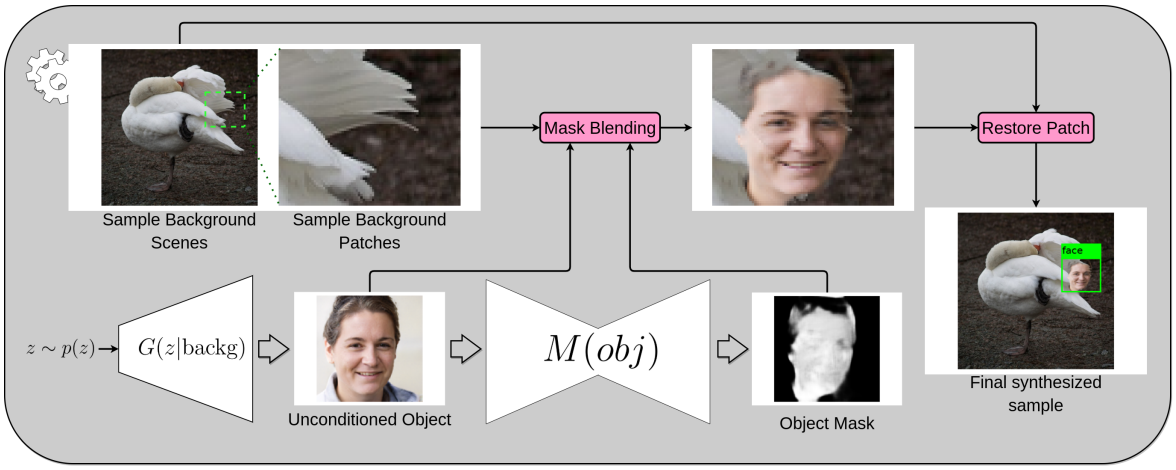


Figure 3.5: *Background Independent Sample Synthesis*

while the generator in traditional GAN architectures has a fixed output size. One trivial solution would be to train several generative models over different input scales, therefore increasing the method's complexity and restricting the possible object scales to a discrete set. However, this method was already unstable with small output sizes, and we believe it would be even harder to make it work on higher resolutions. The only effective way we found to counter this issue was to downscale the conditioned background patches and later upscale the generated objects, further damaging the already poor object image quality.

A second problem we observed in the situations where the method worked is the fact that we could not control the effective size of the objects. By design, the generated objects are confined to the background patches, but the space they occupy inside these patches can vary significantly when using traditional GAN generation. If we simply take the sampled patch's frame as bounding box, we would be introducing significant annotation noise into the synthesized data.

3.1.4 Generating Objects Independently of Backgrounds

These limitations lead us to a second simplification. Instead of generating objects conditioned on background patches, we sample (or generate) the background scenes and generate the objects independently of background. We then sample random patches from the scene in the same way as before and somehow combine them with the objects. Figure 3.5 illustrates this new approach.

By formulating the problem this way, we have the advantage of being capable of reusing already pretrained traditional generative models, which are already available "off-the-shelf" for common object classes, and are usually optimized for object quality. Also, by not having the patch conditioning component, we could follow traditional GAN techniques that are already found to perform stably on high resolution classification data. This way, we could generate objects that are sufficiently big, and then downsample them to any sampled patch size without significant quality loss. Examples of images generated by a modern GAN

Figure 3.6: *Images Generated by a StyleGAN architecture trained on the FFHQ dataset (Karras et al., 2019a).*

architecture (StyleGAN (Karras *et al.*, 2019a)) are displayed in Figure 3.6.

Another advantage, most important for our purposes, is that these traditional generative models are trained with classification level images, which are significantly cheaper to acquire than their detection correspondents, and classification datasets are frequently orders of magnitude bigger than detection datasets. This way, we would be able to take advantage of the knowledge encoded in the cheaper classification data to improve the results of the harder detection problem. This formulation is aligned with our intention of avoiding the need for expensive supervision, while at the same time being partially alignined with the recent trend of unsupervised and semi-supervised pretraining, in which large bodies of unlabeled or weakly labeled data are used to improve results for downstream tasks (see Section 2.4).

The remaining question then is how to combine these classification style object images with the background patches. The ideal way would be to segment the object from the generated background in the GAN synthesized image, so that only the object region is combined with the target patch. The formulation discussed above is shown in Figure 3.5.

This raises a problem, since most of the recent state-of-the-art segmentation techniques are heavily supervised (He *et al.*, 2017; Long *et al.*, 2015; Ronneberger *et al.*, 2015), going against our objective of reducing supervision effort. To avoid the need for expensive supervision, we set out to investigate techniques that could be used to perform unsupervised segmentation, as we briefly describe next.

3.2 Unsupervised Segmentation

Since the adoption of Deep Learning for Computer Vision problems, the supervised paradigm has been the first and most well investigated approach for most problems. This includes both Semantic and Instance Segmentation, where the currently leading techniques require heavy supervision for training, with pixel level annotations in the form of ground truth segmentation masks (He *et al.*, 2017; Long *et al.*, 2015; Ronneberger *et al.*, 2015).

In this perspective, Unsupervised Segmentation consists of the task of mapping images to some form of segmentation output, using architectures or techniques that do not require the segmentation ground truth during training. This is a much harder problem than traditional supervised segmentation, and only recently we had works demonstrating promising results on realistic contexts beyond simple toy problems. As a consequence, the entries in this field are very recent and build upon very different ideas.

In Unsupervised Image Segmentation by Backpropagation, Kanezaki (2018) performed unsupervised segmentation by using an iterative optimization approach. The authors paired a CNN feature extractor with a cluster assignment algorithm over the features extracted for each pixel. They then performed alternate optimizations of the CNN using traditional backpropagation, and of the cluster assignment. This process needs to be done for several iterations in order to segment each image (there is no training step), causing it to be expensive if needed to be used for a big number of images.

Additionally, Unsupervised Segmentation could theoretically be accomplished with "Copy-Pasting" GANs (Arandjelović and Zisserman, 2019), as a byproduct of its Object Discovery task. The goal here is to crop an object from a source image I_s, and paste it onto a destination image I_d in a realistic way. For this, the method uses a generator that outputs a segmentation mask for I_s, that is used for blending it with the target, and a discriminator that tries to classify real images from composed ones. The reasoning here is that, in order to fool the discriminator, the generator would need to learn how to segment full objects in the source image. The authors also needed some additional training tricks in order to prevent certain shortcuts, such as the generator producing a "copy-all" or "copy-nothing" mask. Object Discovery seems a promising direction that could eventually allow advances in Unsupervised Segmentation. However, this technique is still mostly limited to simple, artificial domains (Arandjelović and Zisserman, 2019). Moreover, in order to train a Copy-Pasting GAN, one needs access to images that do not contain the objects of interest, while also following the same overall distribution of backgrounds, in order to prevent the discriminator from focusing on subtle cues instead of the objects themselves.

A technique related to Unsupervised Segmentation that could potentially be used in our method is the Layered Recursive GAN (LR-GAN) (Yang *et al.*, 2017). It is not concerned with segmenting images, but rather with generating images (GAN style) paired with segmentation masks. It does this to generate background and object in sequence. Approaches such as LR-GAN are promising for sample synthesis, as they combine the generation of the

image and the label into a single architecture. But in our attempts, we found LR-GAN to be unstable to train, and the visual sample quality to be bellow that of the technique presented next.

For our method, we opted for the Unsupervised Segmentation by Redrawing (ReDO) method (Chen *et al.*, 2019), which was proposed more recently showing promising results on real world domains. Due to it being "Unsupervised" from the Segmentation perspective, it implies that only classification level annotations are required for training all stages of the method, thus aligning with our motivation for avoiding expensive supervision. Next, we explain how the ReDO method works.

3.2.1 Unsupervised Object Segmentation by Redrawing (ReDO)

The ReDO framework (Chen *et al.*, 2019) works with the assumption that objects in an image can be modeled by a generative process independently from each other. It consists of a network with a branch responsible for segmenting images into a set of disjoint objects + background, with another branch that generates fake objects inside these segmented regions (redrawing), starting from noise inputs. This network is trained adversarially against a discriminator that tries to tell real images apart from the ones that were segmented and then redrawn, following the GAN paradigm (Goodfellow *et al.*, 2014). ReDO takes advantage of the object independence assumption by making the generation (redrawing step) of each object completely independent of what is present in the segmented regions of other objects. As training progresses, the generator learns both to better redraw realistic objects within the segmented regions, as well as to produce more realistic segmentation masks, which is the ultimate goal of the approach.

With this formulation only, the models can take shortcuts to trivial solutions, such as when a single object becomes responsible for representing the whole image, with all the others producing empty masks. In order to prevent these unwanted trivial solutions, Chen *et al.* (2019) use a noise reconstruction network δ, responsible for reconstructing the noise vector used by the redrawing branch, using only its output region, which would be impossible if the segmentation mask were empty. The ReDO method is illustrated in Figure 3.7.

ReDO demonstrated impressive results in a set of real world domains: flowers, faces, and birds. Despite three object classes being relatively little for modern Computer Vision standards, it is still impressive considering that the model is producing pixel level predictions in an unsupervised way. For additional details, please refer to Chen *et al.* (2019).

We choose to adopt ReDO in our experiments mostly because we already knew it could handle a set of (although simple) real world domains, but also because it fits nicely in a pipeline structure. Its end result is not concerned with generating images along with masks, as is done in Yang *et al.* (2017) (although this could also be achieved using the redrawing branch), but rather with taking images as input and segmenting them. This allows us to use it in combination with existing "off-the-shelf" generative image models, already optimized

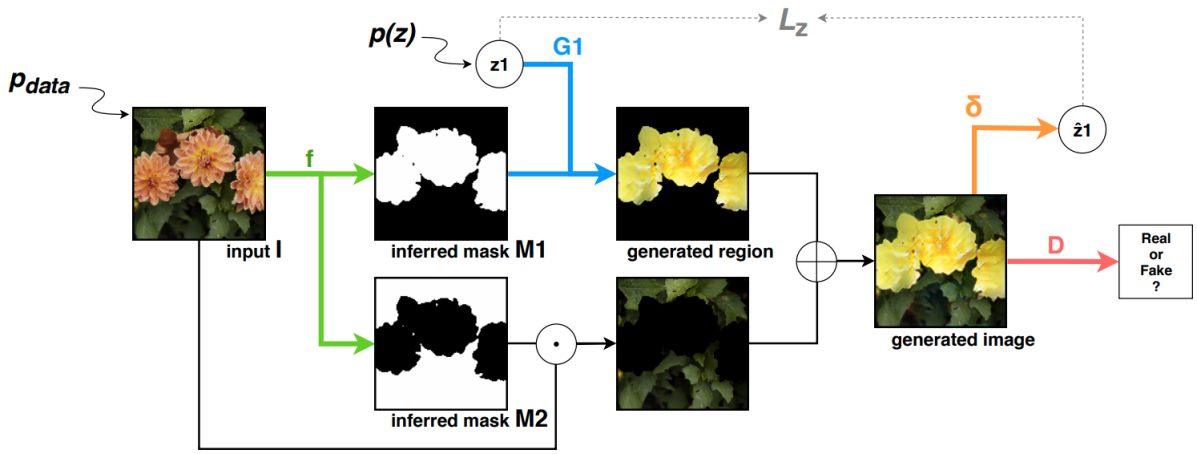


Figure 3.7: *Unsupervised Segmentation by Redrawing. Image taken from Chen et al. (2019) (Figure 1).*

for image quality.

In our attempts, we found the ReDO method to be much easier to train than the LR-GAN (Yang *et al.*, 2017), although we did observe the training diverge and fail a few times, a problem also reported by the authors (Chen *et al.*, 2019). It is plausible to expect Unsupervised Segmentation to mature in the next years, which will lead to better architectural and optimization practices that would make the training easier.

3.3 Proposed Synthesis Pipeline

In this section, we summarize the pipeline of tasks we used to perform object detection sample synthesis for our experiments. This is the outcome of the literature research and practical attempts as described in the past sections. This pipeline is visually described in Figure 3.8, and can be broken down into four steps:

1. **Image Gathering.** The first step consists of gathering classification level image samples. These are traditional, centralized single object images, in the format used for classification tasks. This gathering can be performed either by just sampling real classification images, or by using a traditional generative image model previously trained on such real images.

   In this work, we advocate for using a generative model (Figure 3.8), which we accomplish using standard GAN generation (Goodfellow *et al.*, 2014). In Section 4.3.4 we demonstrate that both options produce equivalent results, and discuss the motivations in favor of this choice. Note that generative models used in this stage are trained only with classification level annotations. Details of the generative models used in our experiments are presented in Chapter 4.

2. **Unsupervised RoI Segmentation.** The next step is to segment the object region of interest in the gathered ("real" or "fake") object images. As described above, since

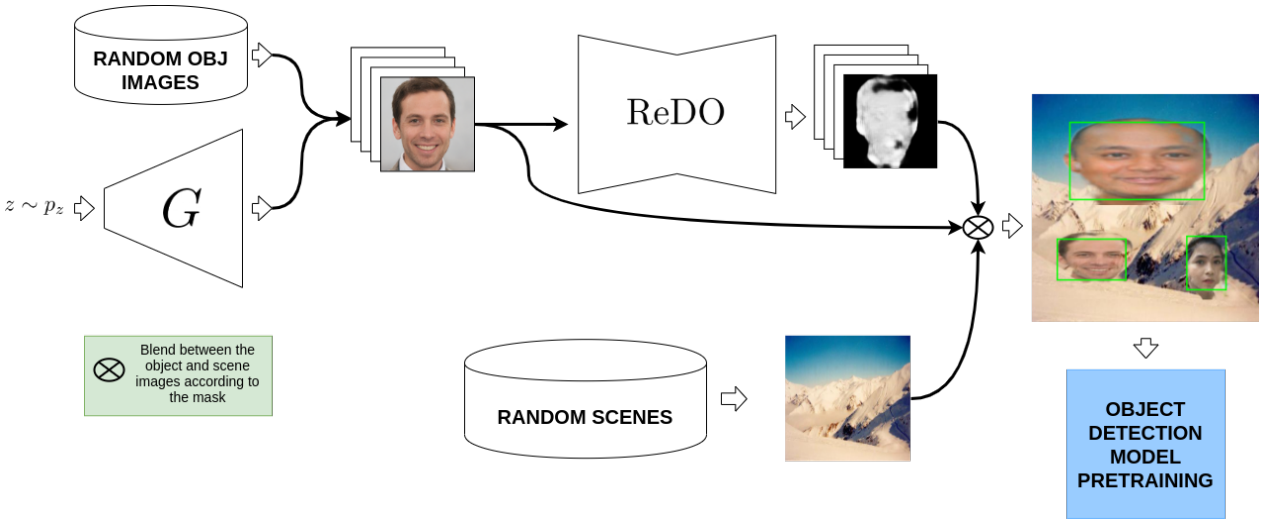


Figure 3.8: *Overview of the synthesis pipeline we adopted for our experiments. We opted for this structure due its simplicity and thanks to recent advances in Unsupervised Segmentation (Chen et al., 2019) having allowed it to be possible. Different formulations can potentially work as well, or even surpass this one.*

we aim to exploit image level classes only, we must restrict ourselves to segmentation techniques that can be trained without mask annotations. In this work, we opted for the above described (Section 3.2.1) Unsupervised Object Segmentation by Redrawing (ReDO) method (Chen *et al.*, 2019).

3. **Object Detection Sample Synthesis.** Once we have generated images of a given object class paired with their segmentation masks, we can create detection samples by placing the segmented object regions on top of random (real or generated) scenes, at varying number and at random positions and scales, through a simple mask merging operation:

$$Op = Go \times M(Go) + Bp \times (1 - M(Go))$$

where Go, Bp, and Op are the generated object image, sampled background patch, and output merged patch, respectively, and $M(Go)$ is the object's segmentation mask. From the masks, we can also easily extract the bounding box parameters.

We point to the fact that, as we choose to use GAN generated classification images instead of real ones in the Image Gathering stage, we can now essentially generate an infinite stream of synthesized images with bounding box annotations, where (almost certainly), no object instance will appear more than once. We also point again to the fact that the only supervision needed for the previous two steps is at the classification level, which allows us to potentially exploit much larger datasets.

4. **Object Detection Pretraining.** Finally, we use this infinite stream of synthesized samples to pretrain object detection models. This pretraining initialization strategy is illustrated in Figure 3.9. We demonstrate in Chapter 4 that initializing a detection

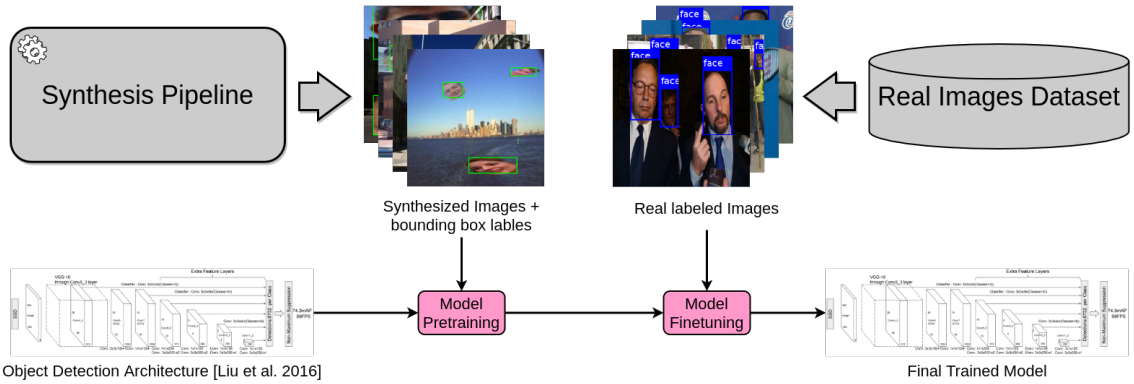


Figure 3.9: *Our Pretraining Initialization Strategy.*

model using this pretraining strategy allows us to achieve comparable results with a significantly smaller amount of real labeled detection images. We also discuss in Section 4.3.2 alternative ways to take advantage of this infinite stream of samples and reasons for why we argue in favor of this pretraining initialization strategy.

We highlight here that the goal of this project is to demonstrate that it is possible to take advantage of the knowledge encoded in cheaper forms of supervision, by using synthesized samples to improve detection results and/or reduce the need for expensive real labeled data. We do not claim this pipeline to be the optimal strategy to perform this synthesis or knowledge transfer. We merely took advantage of existing techniques and trends to allow us to demonstrate this in a systematic way. In particular, we expand upon recent works on the still emerging field of Unsupervised Segmentation, while motivated by Unsupervised and Semi-supervised Pretraining methodologies.

We also do not claim the proposed pretraining initialization to be the optimal way to incorporate these synthesized samples into the traditional model training. However, we do perform an experimental evaluation and offer arguments in favor of this choice in Section 4.3.

To conclude this chapter, we highlight again the ways in which this project improves upon existing attempts. First of all, when considering other Generative Sample Synthesis approaches, while most of them only consider synthesis for classification problems (Antoniou *et al.*, 2017; Mariani *et al.*, 2018; Wang and Perez, 2017; Yamaguchi *et al.*, 2019; Zhang *et al.*, 2019), this project proposes a generative synthesis mechanism for Object Detection, a considerably harder problem. This required us to find clever ways of combining existing techniques. Second, to the best of our knowledge, among the works that do attempt to synthesize images for problems beyond Image Classification, ours is the first one that do so without needing complex forms of supervision, such as segmentation masks (Bailo *et al.*, 2019; Bowles *et al.*, 2018; Milz *et al.*, 2018), paired images from different domains (Sandfort *et al.*, 2019), or manually designed simulators (Shrivastava *et al.*, 2017).

Chapter 4

Experiments

In this chapter, we present the experimental evaluation conducted, as well as a discussion about the results obtained. Overall, these experiments were designed in order to evaluate the following aspects:

- How much does this Synthesized Samples pretraining help in the Object Detection problem?
- In particular, how much better are the results when using little real labeled data? This is a critical aspect if we want this technique to be useful as a way to reduce the dependency on expensive supervision.
- How good would the results be if we had only synthesized samples? Or in other words, how necessary real labeled samples still are?
- What is the influence of this pretraining followed by finetuning strategy? Could we get equivalent results by simply training on real and synthesized samples mixed together? What are the advantages of each approach?
- Regarding our synthesis pipeline. How much influence does the segmentation step has on the results? Could we get similar results by just pasting full classification level images on top of background scenes?
- Are we loosing some performance by using GAN generated classification images at the first pipeline stage? Could we get better results by using real classification samples? What are the advantages of each approach?
- What happens if we use this pretraining initialization strategy with datasets that are already very big? Also, what happens when the synthesized samples look too different from the real data?

First, we perform experiments in the relatively simple and artificial domain of QR Code detection, without including all stages of our method, in order to check our hypothesis that synthesized samples can improve detection results and/or data efficiency, while using a more restrained context (Section 4.1). Next, we investigate the performance of the full method

on more realistic domains. In particular, we apply it for detection of faces, birds, and cars (Section 4.2), and perform a series of ablation experiments in order to evaluate the influence of different design choices (Section 4.3). Finally, we conduct an additional set of experiments using the large scale face detection dataset WIDER Face (Yang *et al.*, 2016), in order to observe how the proposed method behaves in a situation where data is already abundant, and the synthesized and real data distributions differ substantially (Section 4.4).

Unless otherwise noted, the experiments that use our pretraining initialization strategy use GAN generated classification images for the first stage of the synthesis pipeline (the *Image Gathering* step). In Section 4.3.4 we investigate the implications for this choice.

With the exception of external implementations, all the experiments were implemented using TensorFlow (Abadi *et al.*, 2015). The code to reproduce these experiments has been made publicly available on https://github.com/Leonardo-Blanger/synthesis_pretraining_object_detection.

4.1 Preliminary Results on QR Code Detection

For this set of experiments, we adopted a QR Code detection dataset, which we labeled for a previous work[1] (Blanger and Hirata, 2019). It is a relatively small dataset for deep learning standards, although big enough for the task's simplicity. It is composed of 567 training and 100 test images. For these experiments, we additionally reserved 100 images from the training partition to be used as validation set. We used this validation set to perform early-stopping, as we will detail next. Labeled samples from the training set are displayed in Figure 4.1.

To synthesize detection samples, we used the `python-qrcode` library[2] to generate traditional black on white QR Code images. These codes are generated with a version sampled uniformly between 1 and 10 (the code version determines, among other things, the size of the code matrix), and encode random lowercase text strings with length sampled uniformly between 1 and 100.

By using this library, we avoid training a generative image model for QR Codes. As we have full control over the generated code frame, we also get the code masks for free. With this formulation, we can synthesize detection samples without the neural network components of the pipeline.

We used samples from the Pascal VOC 2007 and 2012 datasets (Everingham *et al.*, 2007, 2012) as background scenes, and applied a series of simple geometric and photometric transformations to the code images before pasting them onto the scenes (with the geometric ones being mirrored on the masks), taken from the default demo of the `imgaug` library[3] (Jung *et al.*, 2020). We randomly sampled the number of codes per image uniformly

[1] https://github.com/ImageU/QR_codes_dataset
[2] https://github.com/lincolnloop/python-qrcode
[3] https://imgaug.readthedocs.io/en/latest/source/examples_basics.html

Figure 4.1: *Samples from the training partition of our QR Codes dataset (Blanger and Hirata, 2019) with boudning box annotations.*

between 1 and 10. And for each code, we sampled its scale within the image uniformly between 5% and 80%, its position uniformly within the whole image frame, and its aspect ratio uniformly between 1/2 and 2. Examples of these synthesized samples are displayed in Figure 4.2.

We performed experiments using the Single Shot Detection model (SSD) (Liu *et al.*, 2016), with vs without our pretraining initialization strategy. We used both a MobileNet (Howard *et al.*, 2017) and a ResNet50 (He *et al.*, 2016) backbone CNNs, with four and six detection scales, respectively. More details of the architectures are available in Appendix B. For simplicity, we resized all input images to 300×300. We used the Adam optimizer (Kingma and Ba, 2014), with learning rate 10^{-4}, and first and second moment decay rates of 0.9 and 0.999, respectively. We optimized the Focal Loss (Lin *et al.*, 2017) and Smooth L1 loss (Girshick, 2015) for the classification and box regression output branches of the SSD, respectively. We used a batch size of 16 images. For fairness and simplicity, we did not perform any form of search over the overall architecture, backbone, or training hyper-parameters.

We report results using Mean Average Precision at 0.5 IOU (mAP@0.5), following the Pascal VOC competition metric (Everingham and Winn, 2011). For details of the mAP metric in the context of Object Detection, see Appendix A.

The models with our pretraining initialization strategy were first trained on the stream

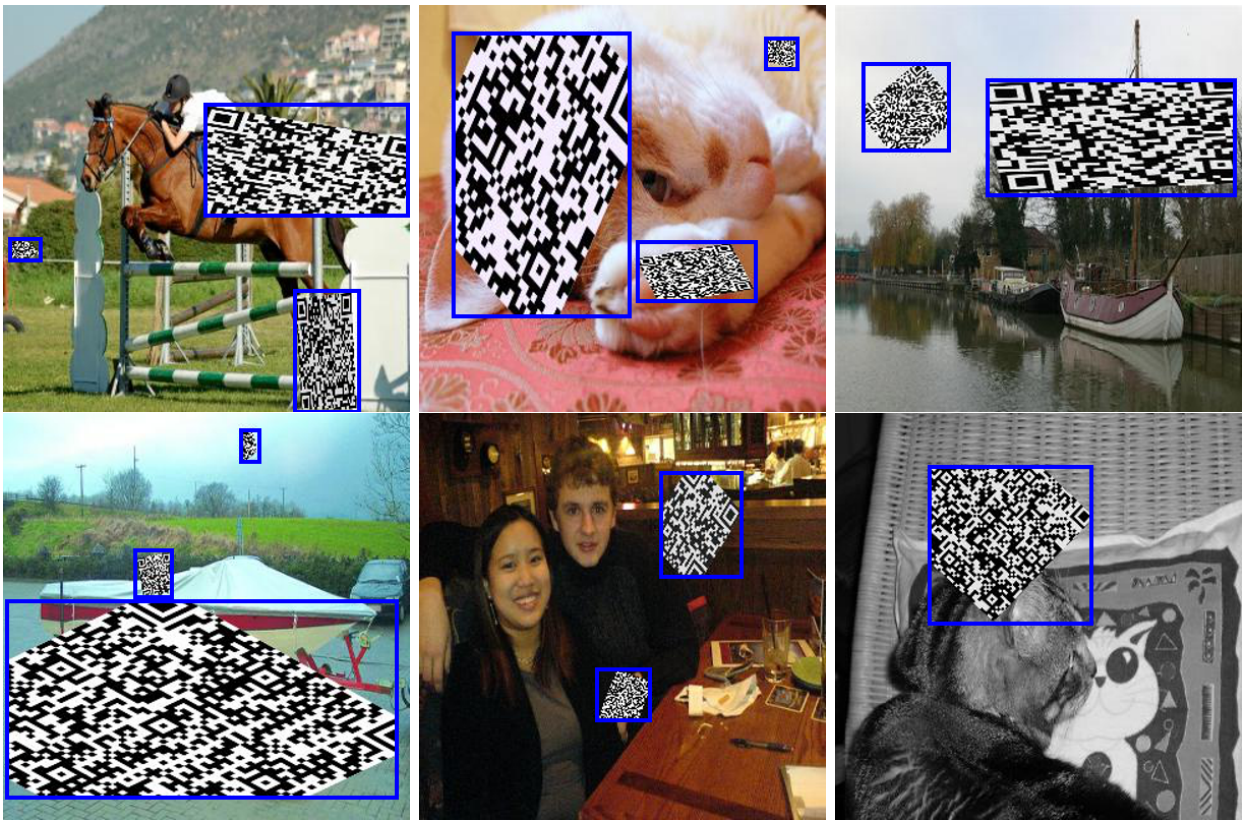


Figure 4.2: *Synthesized samples with bounding box annotations.*

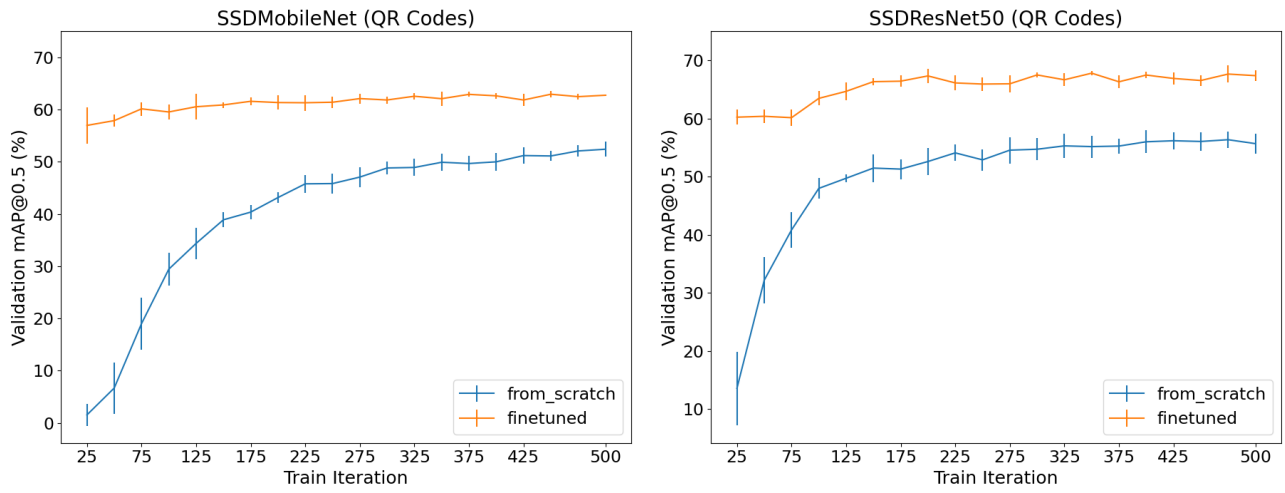
of synthesized samples for 500 iterations, checkpointing and applying the model to the held out validation set every 25 iterations. We used the best performing of these checkpoints as the "final" pretrained version. The models without our initialization strategy had traditional initialization instead, with ImageNet (Deng *et al.*, 2009) classification weights for the backbone CNN and uniform Glorot/Xavier initialization (Glorot and Bengio, 2010) for the detection specific layers.

We designed these experiments around two aspects. First, we wanted to identify how this pretraining initialization affected the final detection results. Second, we wanted to understand how this effect changes when we consider different amounts of real samples for finetuning.

Initially, we finetuned the models with all available real data, with vs without our initialization strategy. For each run, we again trained for 500 iterations, and followed the same checkpointing and validation every 25 iterations. We considered three independent runs. Figure 4.3 shows the validation mAP during training.

As we can see, the pretrained versions achieve a significantly higher mAP on the validation set. At the same time, they are capable of converging faster to a solution, and behave more stably across independent runs, as demonstrated by the shorter deviation bars. Also noticeable, the 500 iterations we choose are enough for convergence even in this case where we use all the available real data.

Next, we trained a series of models using different amounts of real samples, with vs



(a) **(b)**

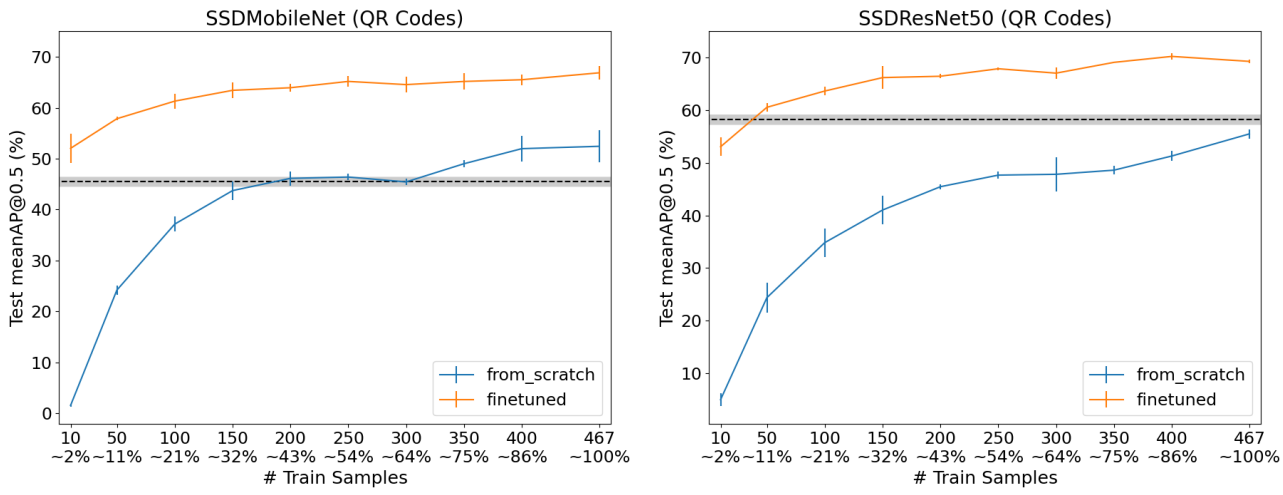
Figure 4.3: *Results on the QR Codes validation set during training with vs without pretraining, of an SSD detection model with MobileNet (a) and ResNet50 (b) backbone CNNs, in terms of mAP at 0.5 IOU, when using 100% of the real data. Values are averages with deviation bars taken from three independent runs with the same training configurations.*

without our pretraining strategy. For each quantity of samples, we used the same subset of the real data for both versions and across all architectures and runs. From each model, we again trained for the same 500 iterations with checkpointing and validation every 25 iterations as before, and choose the checkpoint with the best validation mAP as the final model version, in effect performing early-stopping. We then evaluated the final version of each model on the test set, and the results are shown in Figure 4.4.

The horizontal axis in Figure 4.4 represents the number of real images that were used for finetuning. This images were independently resampled for each quantity. The vertical axis shows the mean Average Precision on the test set, computed for the best validation checkpoint of each model version after training. The values plotted are averages over three runs, which use the same subset of training samples, while the vertical bars present the standard deviations over these runs. The blue and orange curves present the results for the models trained "from scratch" and "finetuned", that is, without vs with our pretraining initialization strategy. The horizontal dashed line is the test set results for models right after being pretrained on synthesized samples, but before being finetuned on real samples (more details in Subsection 4.3.1).

For both architectures, the version that was pretrained on synthesized samples achieves a significant advantage with respect to all amounts of real samples used for finetuning. Moreover, the difference is more noticeable when considering small amounts, supporting our hypothesis that initializing the model with synthesized samples allows us to achieve comparable results with significantly less real labeled data. Additionally, they again behave more stably across independent runs.

An interesting phenomenon happens for the ResNet50 backbone when using too few real samples (10 images, or around 2%). The test set mAP was actually higher for the model



(a) **(b)**

Figure 4.4: *Results on the QR Codes test set for the final model versions with vs without pretraining, on an SSD detection model with MobileNet (a) and ResNet50 (b) backbone CNNs, in terms of mAP at 0.5 IOU. Values are averages with deviation bars taken from three independent runs with the same training configurations. The horizontal dashed lines represent the average mAP of the initialized model (after pretraining on synthesized samples but before being finetuned on real data) with the gray margin representing the deviation, also considering three independent runs.*

right after initialization on the synthesized samples, than when it was finetuned on this small number of real images. We hypothesize this to be due to overfitting. The ResNet50 backbone, being a more powerful model than the MobileNet, might be memorizing these 10 samples before the first validation checkpoint has the chance to capture a more generalizable version of the parameters.

4.2 Main Results

In this section, we present results on three more realistic scenarios: face, car and bird detection, all of which require all the stages of our synthesis pipeline. Bellow we describe the datasets we used for each of these sets of experiments.

- **Faces.** We used the *Face Detection Data Set and Benchmark* (FDDB) (Jain and Learned-Miller, 2010). This dataset contains 5171 faces across 2845 images that are divided into 10 roughly equal sized folds. We used the first five folds for training, the next two for validation and the last three for testing, making for a split of 1449/581/815 images. We converted the face ellipsis labels to rectangular bounding boxes [4].

  In order to generate fake face images, we used StyleGAN faces (Karras *et al.*, 2019a), which was pretrained on the FFHQ dataset (Karras *et al.*, 2019a), and the ReDO segmentation branch with weights from the Labeled Faces in the Wild dataset

[4] We used the script provided in github.com/ankanbansal/fddb-for-yolo

(LFW) (Huang *et al.*, 2007; Learned-Miller, 2014), as provided by Chen *et al.* (2019)[5].

- **Birds.** We used the *Caltech-UCSD Birds-200-2011* (CUB) dataset (Wah *et al.*, 2011). This dataset contains 11788 images, each with a single bird of varying scale. We used a DM-GAN (Zhu *et al.*, 2019) generator to generate the fake bird images and the ReDO segmentation branch provided by Chen *et al.* (2019), both of them trained on the bounding box crops of the CUB dataset itself.

  We point to the fact that neither the DM-GAN (Zhu *et al.*, 2019) nor the ReDO segmentation model (Chen *et al.*, 2019) were trained following the official train/test splits of the CUB dataset (Wah *et al.*, 2011). Therefore, in order to prevent leakage of test information on the first stages of our synthesis pipeline, we only used the intersection between the test sets from Zhu *et al.* (2019) and Chen *et al.* (2019) as our test set, which totals 443 images. We additionally picked 1000 images from the remaining ones as validation, making for a 10345/1000/443 split.

- **Cars.** We used the Stanford Cars dataset (Krause *et al.*, 2013), which contains 16185 images, each with a single car of varying scale. We picked 1000 images from the official train split to use as validation, making for a 7144/1000/8041 split.

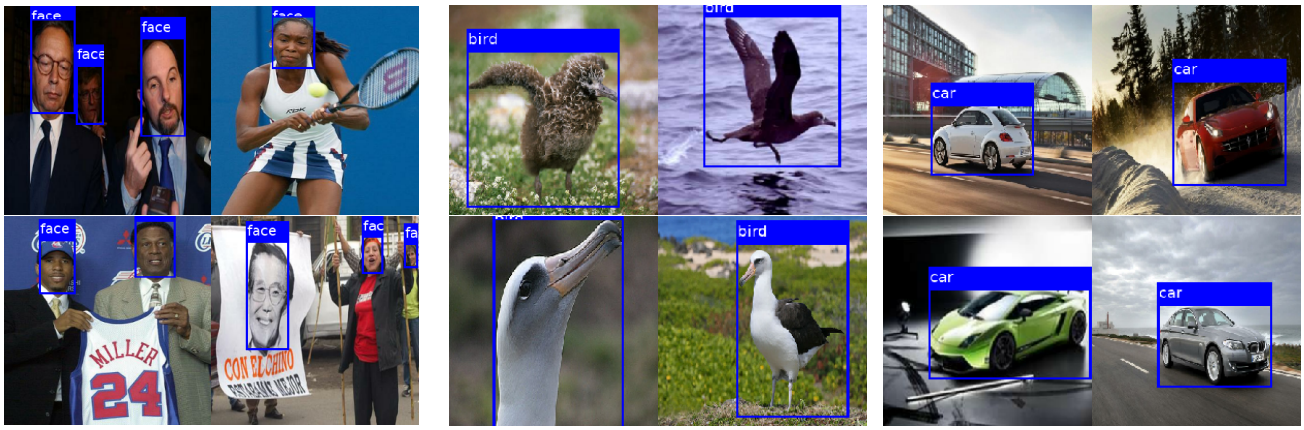
  To generate fake car images, we used StyleGAN Cars (Karras *et al.*, 2019a), with weights from the LSUN Cars dataset (Yu *et al.*, 2015). As Chen *et al.* (2019) did not provide weights for cars, we trained a full ReDO segmentation model on the fake images just mentioned using the same model configurations as for the faces.

For all three domains, we again applied the same set of `imgaug` random color and geometric augmentations to the fake images (Jung *et al.*, 2020), with the geometric ones being mirrored on the respective masks. We again sampled images from the Pascal VOC 2007 and 2012 datasets (Everingham *et al.*, 2007, 2012) to use as background scenes. For the faces and cars, we ignored the Pascal images that contained objects from the classes "Person" and "Car", respectively, in order to avoid having unlabeled objects in the background for the synthesized samples.

We sampled the number of instances of each object uniformly at random from the ranges [1, 10] for faces, [1, 3] for birds, and [1, 5] for cars. We sampled the object scale uniformly from the ranges [5%, 80%] for faces, [50%, 90%] for birds, and [50%, 100%] for cars. We sampled the object aspect ratio uniformly from the ranges [1/2, 2] for faces and birds, and [1/3, 3] for cars. For all objects, we sampled the object position uniformly over the whole image, taking care to prevent multiple instances from covering each other. We choose these parameters in such a way as to make the resulting synthesized samples seem plausible in comparison to the respective real world datasets.

Figure 4.5 shows some samples of real and synthesized images with their bounding box labels. We note that the samples are far from realistic, both from a global perspective, as

[5] github.com/mickaelChen/ReDO



(a)

(b)

Figure 4.5: ***(a)*** *Real samples with bounding box annotations from the FDDB Faces (Jain and Learned-Miller, 2010) (left), CUB Birds (Wah et al., 2011) (middle), and Stanford Cars (Krause et al., 2013) (right) datasets.* ***(b)*** *Synthesized samples for each of the three domains.*

the synthesis pipeline has no form of background conditioning to produce coherent object placements (faces in the sky and huge birds on bicycles do not happen in normal situations), as well as from a local context (by zooming in, one can see unrealistic texture artifacts and transparencies). Nonetheless, we experimentally demonstrate that an infinite stream of these cheap synthesized samples can reduce the need for real labeled data, without requiring any other form of more expensive supervision.

We performed these experiments using the same set of models as for the QR Codes case (Section 4.1), with the same training configurations, and the same strategy to select the pretrained version for each model. The only difference being that we trained each model for 1000 iterations in the case of birds, both for initialization with the synthesized samples, as well as for the normal training on real data, instead of 500 as is done for the other domains.

To start with, we again trained the models with all available real data, with vs without our initialization strategy. Figure 4.6 shows the performance on the validation sets during training for the three domains.

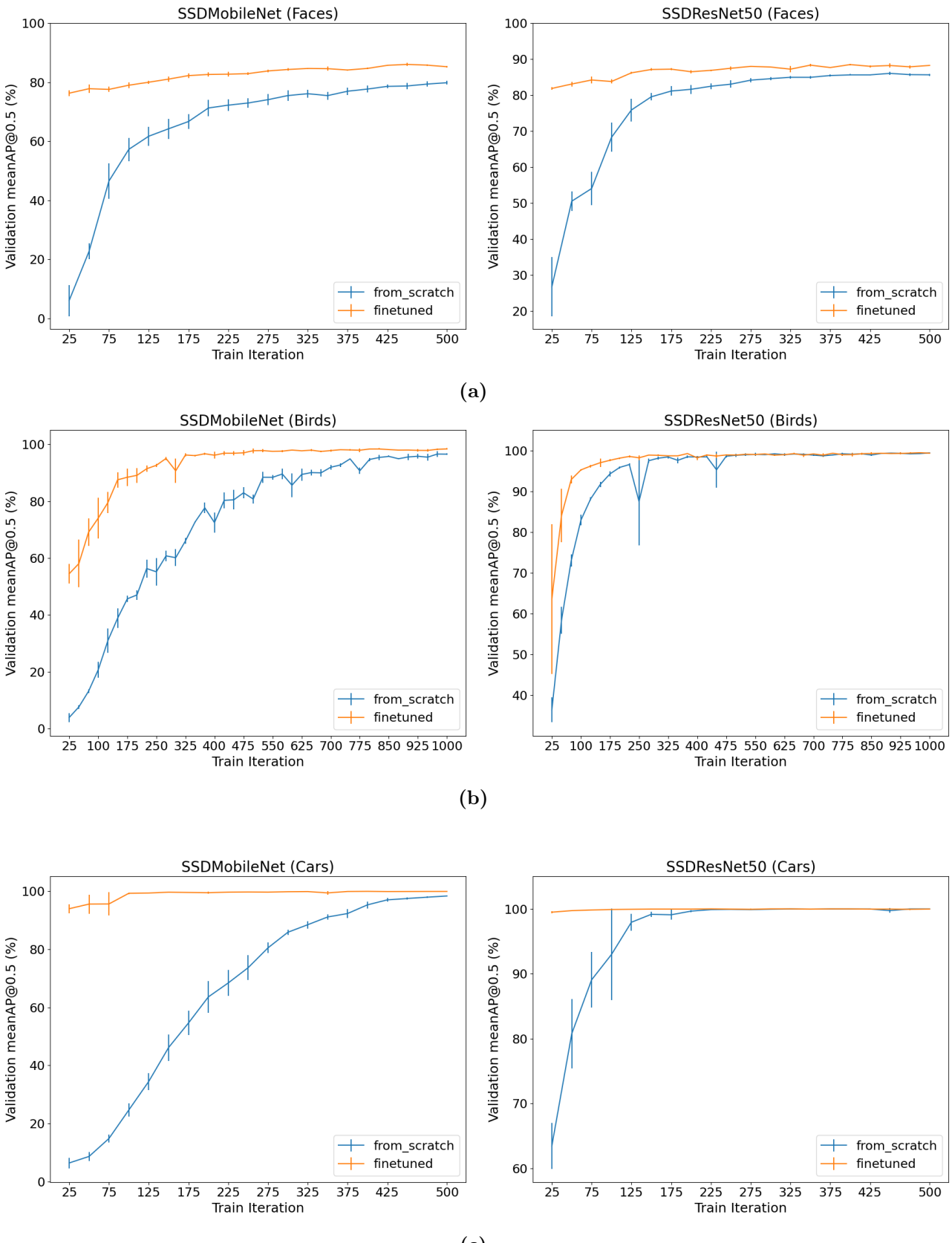


Figure 4.6: *Results on the Faces (a), Birds (b), and Cars (c) validation sets during training with vs without pretraining, on an SSD detection model with MobileNet (left) and ResNet50 (right) backbone CNNs, in terms of mAP at 0.5 IOU, when using 100% of the real data. Values are averages with deviation bars taken from three independent runs with the same training configurations.*

Overall, we observe the same patterns as when training the model on QR Codes. The model that was pretrained with synthesized samples had a faster convergence in all domains and architectures, and once converged, it never performed worse than with the traditional initialization, performing considerably better in some situations (see Faces in Figure 4.6a).

Next, we trained a series of models using different amounts of real samples, with vs without our pretraining strategy. Again, for each quantity of samples, we used the same subset of the real data for both versions and across all architectures and runs. From each model, we again trained for the same 500 iterations in the case of Faces and Cars, and 1000 iterations in the case of Birds, with checkpointing and validation every 25 iterations as before. We choose the checkpoint with the best validation mAP as the final model version, in effect performing early-stopping. We then evaluated the final version of each model on the test set, and the results are shown in Figure 4.7. These graphs are structured the same way as for the QR Codes in Figure 4.4.

These results were similar to the ones observed for the QR Codes case. Once again, the models that were pretrained never perform worse than the initialized ones, and in several situations perform significantly better. We noticed that, depending on the type of object, the models that were not pretrained can close the gap if enough real samples are provided. We can observe this on case of Birds and Cars, where the training datasets are relatively big and are not considered particularly difficult tasks. We also notice that the speed with which the gap is closed varies across different backbones.

But most importantly, the advantage of the pretrained versions is more noticeable, and always present, when considering very few real samples, further supporting our hypothesis that initializing the model with synthesized samples reduces the need for real labeled data. We also observe more stable results across runs for the majority of situations.

Tables 4.1 and 4.2 show numeric results for a selected subset of evaluated amounts of real samples, for the MobileNet (Howard *et al.*, 2017) and ResNet50 (He *et al.*, 2016) CNN backbones, respectively. Column 2 indicates how many real samples were used for each row, Columns 3 and 4 refer to the results achieved without and with our pretraining initialization strategy, respectively. Results are presented in mAP@0.5 as averages $\pm$ standard deviations over runs. For the full tables containing results for all evaluated amounts of samples, please refer to Appendix C.

With these tables, it becomes easier to quantify the extent to which the pretraining initialization improves detection results. For instance, with amounts of real samples that range from $\sim 1-2\%$, the advantage of pretrained models when considering the different object classes was of 50.46%, 44.48%, 51.26%, and 55.55%, for the MobileNet backbone (Howard *et al.*, 2017), and of 48.06%, 52.05%, 10.92%, and 14.24%, for the ResNet50 backbone (He *et al.*, 2016).

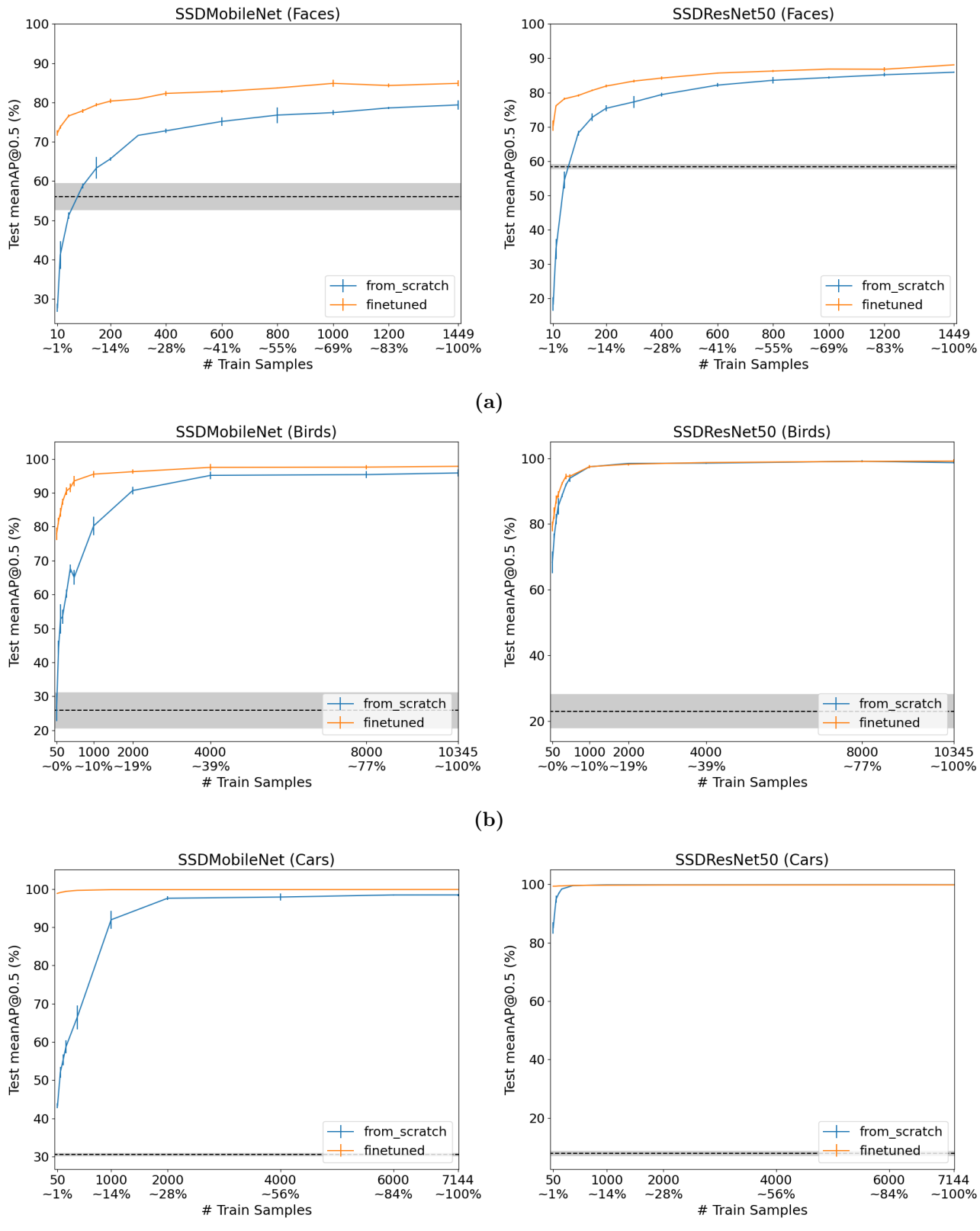


Figure 4.7: *Results on the Faces (a), Birds (b), and Cars (c) test sets for the final model versions with vs without pretraining, on an SSD detection model with MobileNet (left) and ResNet50 (right) backbone CNNs, in terms of mAP at 0.5 IOU. Values are averages with deviation bars taken from three independent runs with the same training configurations. The horizontal dashed line represents the average mAP of the initialized model (after pretraining on synthesized samples but before being finetuned on real data) with the gray margin representing the deviation, also considering three independent runs.*

	# real samples	without pretraining	with pretraining
QR Codes	pretrain only (0%)	–	45.45% ± 0.88%
	10 (∼2%)	1.57% ± 0.30%	52.03% ± 2.91%
	50 (∼11%)	24.20% ± 0.92%	57.84% ± 0.34%
	100 (∼21%)	37.15% ± 1.48%	61.29% ± 1.45%
	250 (∼54%)	46.40% ± 0.64%	65.17% ± 1.05%
	467 (100%)	52.43% ± 3.10%	66.86% ± 1.34%
Faces	pretrain only (0%)	–	56.03% ± 3.35%
	10 (∼1%)	27.73% ± 1.02%	72.21% ± 0.74%
	100 (∼7%)	58.73% ± 0.62%	77.86% ± 0.46%
	200 (∼14%)	65.63% ± 0.41%	80.34% ± 0.54%
	800 (∼55%)	76.78% ± 2.01%	83.68% ± 0.13%
	1449 (100%)	79.36% ± 1.22%	84.86% ± 0.80%
Birds	pretrain only (0%)	–	25.83% ± 5.21%
	50 (<1%)	26.69% ± 4.09%	77.95% ± 1.82%
	100 (∼1%)	44.30% ± 2.08%	81.68% ± 0.99%
	200 (∼2%)	53.42% ± 2.12%	87.35% ± 0.92%
	4000 (∼39%)	95.12% ± 1.06%	97.48% ± 0.97%
	10345 (100%)	95.83% ± 1.09%	97.78% ± 0.09%
Cars	pretrain only (0%)	–	30.51% ± 0.32%
	50 (∼1%)	43.28% ± 0.67%	98.83% ± 0.17%
	100 (∼1%)	52.06% ± 1.43%	99.05% ± 0.10%
	200 (∼3%)	58.71% ± 1.70%	99.34% ± 0.11%
	4000 (∼56%)	97.86% ± 0.95%	99.80% ± 0.03%
	7144 (100%)	98.42% ± 0.28%	99.83% ± 0.01%

Table 4.1: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with a few representative numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# real samples	without pretraining	with pretraining
QR Codes	pretrain only (0%)	–	58.20% ± 0.89%
	10 (∼2%)	5.00% ± 1.27%	53.06% ± 1.77%
	50 (∼11%)	24.38% ± 2.87%	60.56% ± 0.81%
	100 (∼21%)	34.83% ± 2.73%	63.65% ± 0.81%
	250 (∼54%)	47.67% ± 0.62%	67.89% ± 0.23%
	467 (100%)	55.50% ± 0.87%	69.29% ± 0.30%
Faces	pretrain only (0%)	–	58.40% ± 0.66%
	10 (∼1%)	18.33% ± 2.01%	70.38% ± 1.53%
	100 (∼7%)	68.15% ± 0.72%	79.14% ± 0.25%
	200 (∼14%)	75.40% ± 0.88%	81.88% ± 0.41%
	800 (∼55%)	83.57% ± 0.92%	86.26% ± 0.39%
	1449 (100%)	85.90% ± 0.13%	88.06% ± 0.11%
Birds	pretrain only (0%)	–	22.98% ± 5.09%
	50 (<1%)	68.28% ± 3.34%	79.20% ± 1.43%
	100 (∼1%)	76.35% ± 0.78%	83.30% ± 1.74%
	200 (∼2%)	85.36% ± 2.49%	89.01% ± 1.05%
	4000 (∼39%)	98.51% ± 0.25%	98.70% ± 0.07%
	10345 (100%)	98.67% ± 0.28%	99.14% ± 0.55%
Cars	pretrain only (0%)	–	7.82% ± 0.72%
	50 (∼1%)	85.11% ± 1.98%	99.35% ± 0.02%
	100 (∼1%)	94.92% ± 1.26%	99.36% ± 0.07%
	200 (∼3%)	98.39% ± 0.17%	99.46% ± 0.00%
	4000 (∼56%)	99.86% ± 0.01%	99.79% ± 0.02%
	7144 (100%)	99.87% ± 0.01%	99.82% ± 0.01%

Table 4.2: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with a few representative numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

4.3 Ablation Experiments

In this section, we present some experiments we conducted in order to understand the influence of several design choices in our synthesis pipeline.

4.3.1 Importance of Finetuning on Real Data

As we are training using a combination of an infinite stream of synthesized samples, together with a comparably small set of real data, it is natural to ask how much of the observed results can be explained by the real data. In other words, the question is by what extent are the real labeled samples still necessary, and whether or not synthesized samples alone are enough.

To answer this question we go back to the previously shown Figures 4.4 and 4.7 and Tables 4.1 and 4.2. They show evaluation results of the models on the test set just after pretraining them on the infinite stream of synthesized samples, but before finetuning on the real data (dashed lines with gray margins in Figures 4.4 and 4.7 and first line of each block in Tables 4.1 and 4.2). As can be seen, the pretraining initialization alone performs badly, barely breaking the 50% mark for most situations.

This is an evidence that, despite promising steps towards reducing the need for expensive supervision, real labeled data is still very important in order to achieve competitive results. We hypothesize this to be the case due to the difference between the distributions of real and synthesized images. Although the pretraining is able to extract useful knowledge about the overall structure and frequent textures of objects, it is not enough to completely bridge the gap to the real world data, and thus performs poorly on the test set.

We note here that it may be the case that, with more realistic detection samples, the need for real data would diminish. However, in order to properly investigate this possibility, we would need a synthesis mechanism that produces more realistic samples. With our current formulation, the major source of lack of realism is the segmentation step, so techniques in the field of Unsupervised Segmentation would need to improve in order for us to achieve more realistic detection samples. Another possibility is to design better synthesis mechanisms that do not require segmentation. It may be possible that this could be achieved in the future through some clever GAN based design.

4.3.2 Importance of the Pretraining Initialization

We opted for doing this pretraining initialization followed by finetuning on real samples, due to the simplicity of having an infinite stream formulation, and also due to inspiration from already successful techniques on Unsupervised/Semi-supervised pretraining, in which the large volume of "cheap" data is used first, and the smaller "good" data is saved for the end. However, one could argue that these synthesized detection images could be used in a similar way to traditional image augmentation, by mixing them together with the real data

on a single training run. In this section, we provide experimental evidence supporting our choice.

We trained models considering only a small set of real examples, as this is where the use of synthesized samples shows the most advantage. We compared our pretraining initialization strategy against a single training session using a mix of real and synthesized samples. For a fair comparison, and in order to compensate for an eventual "warming-up" effect in the pretrained model, we trained the mixed data models for the sum of the number of iterations in the pretraining and finetuning: 1000 steps in the case of faces and cars, and 2000 steps for birds. As we can not use an infinite stream of synthesized samples in the mixed data case, otherwise the influence from synthesized samples would overrun that of the real data, we performed these experiments using different proportions of synthesized samples. For each proportion, we used the same synthesized samples for both the pretrained and mixed data cases. Results are shown in Tables 4.3 and 4.4. Column 1 indicates the number of real samples used for finetuning for each type of object, while column 2 indicates the number of synthesized samples (which is varied for each type of object). Columns 3 and 4 present the results achieved using the single mixed data training session, and the pretraining followed by finetuning strategies, respectively.

In all cases, the pretraining initialization approach performs better than training with mixed data, but regardless of that, both of them are either matched or surpassed by the infinite stream approach. The difference in the results however, may vary significantly across different domains, ranging from around 20% for QR Codes (both architectures) to only 2% for Cars (with the ResNet50). We believe the model's behavior here depends on the quality of the synthesized samples, or in other words, how similar are the real and synthesized distributions. For instance, in the case of birds, for which the segmentation masks were the worst (see the images in Chen *et al.* (2019)) the gap is the largest. However, further investigation would be required to better understand the nature of this effect.

Nonetheless, as all versions were either matched or surpassed by the infinite stream pretraining approach, we opted for it, as it provides a series of additional practical advantages. Not having to specify the amount of synthesized samples beforehand and not having to worry about sample proportions during batch loading reduces the need for hyper-parameter tuning, and thus also simplify our setup.

4.3.3 Importance of proper Object Segmentation

Next, we set out to investigate the influence of the unsupervised segmentation step. The goal here is to understand if we really need properly cropped objects to synthesize our samples, or we could simply use full classification images instead. For this, we trained a set of models following our pretraining initialization strategy, but when synthesizing the samples, we did not segment the objects, and instead used the whole classification image frame as bounding box. Note that this is essentially what is already being done for the QR

	# fake samples	mixed data	pretr. + finetune
QR Codes 50 real samples (∼10%)	50 (1×)	24.97% ± 1.95%	42.13% ± 1.77%
	100 (2×)	27.41% ± 1.92%	42.33% ± 1.12%
	200 (4×)	26.94% ± 1.49%	45.88% ± 3.04%
	400 (8×)	26.23% ± 1.75%	46.22% ± 1.18%
	inf. stream	–	**57.84% ± 0.34%**
Faces 100 real samples (∼7%)	100 (1×)	68.67% ± 1.32%	73.36% ± 1.00%
	200 (2×)	68.45% ± 1.87%	74.74% ± 0.16%
	400 (4×)	68.23% ± 0.86%	75.09% ± 0.37%
	800 (8×)	69.11% ± 1.52%	76.29% ± 0.19%
	inf. stream	–	**77.86% ± 0.46%**
Birds 100 real samples (∼1%)	100 (1×)	31.54% ± 4.15%	39.89% ± 1.98%
	200 (2×)	31.53% ± 4.78%	45.39% ± 1.95%
	400 (4×)	31.13% ± 2.69%	45.54% ± 2.11%
	800 (8×)	30.25% ± 2.47%	47.65% ± 1.53%
	inf. stream	–	**81.68% ± 0.99%**
Cars 100 real samples (∼1%)	100 (1×)	76.38% ± 1.01%	91.68% ± 1.87%
	200 (2×)	76.46% ± 2.87%	92.22% ± 1.14%
	400 (4×)	76.26% ± 1.18%	94.50% ± 0.79%
	800 (8×)	79.27% ± 2.13%	95.92% ± 0.35%
	inf. stream	–	**99.05% ± 0.10%**

Table 4.3: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with varying amounts of synthesized samples, using a mixed data training session vs our pretraining initialization strategy. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# fake samples	mixed data	pretr. + finetune
QR Codes 50 real samples (∼10%)	50 (1×)	28.13% ± 1.93%	38.71% ± 0.60%
	100 (2×)	27.45% ± 2.17%	43.08% ± 0.96%
	200 (4×)	24.90% ± 0.91%	43.09% ± 3.03%
	400 (8×)	27.72% ± 1.89%	47.46% ± 0.75%
	inf. stream	–	**60.56% ± 0.81%**
Faces 100 real samples (∼7%)	100 (1×)	72.96% ± 0.18%	77.06% ± 0.38%
	200 (2×)	72.59% ± 0.89%	77.56% ± 0.30%
	400 (4×)	73.19% ± 0.26%	77.82% ± 0.30%
	800 (8×)	72.19% ± 0.67%	**79.19% ± 0.30%**
	inf. stream	–	**79.14% ± 0.25%**
Birds 100 real samples (∼1%)	100 (1×)	57.94% ± 0.85%	62.29% ± 2.56%
	200 (2×)	59.49% ± 0.36%	62.52% ± 0.33%
	400 (4×)	58.06% ± 1.78%	66.32% ± 3.52%
	800 (8×)	60.15% ± 0.72%	65.24% ± 3.11%
	inf. stream	–	**83.30% ± 1.74%**
Cars 100 real samples (∼1%)	100 (1×)	96.91% ± 0.06%	98.73% ± 0.13%
	200 (2×)	96.93% ± 0.24%	98.88% ± 0.10%
	400 (4×)	96.79% ± 0.46%	99.08% ± 0.10%
	800 (8×)	96.83% ± 0.11%	98.93% ± 0.02%
	inf. stream	–	**99.36% ± 0.07%**

Table 4.4: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with varying amounts of synthesized samples, using a mixed data training session vs our pretraining initialization strategy. Each value is the average with standard deviation from three independent runs with the same training configurations.*

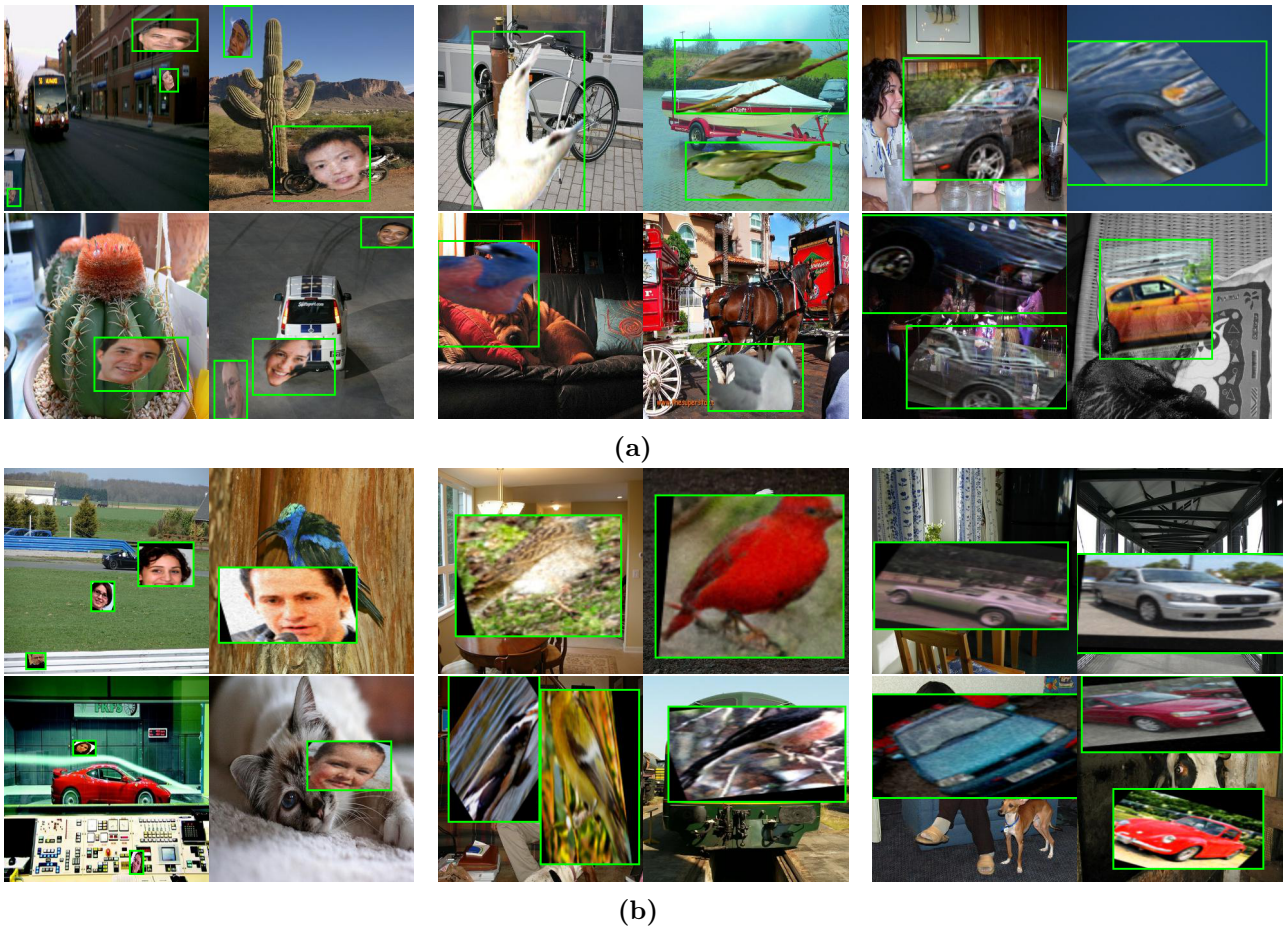


Figure 4.8: *Visual comparison between synthesized samples created with the ReDO segmentation stage (a) and using the naive pasting strategy (b).*

Codes case, so we only performed experiments for the other types of objects. Figure 4.8 compares instances of synthesized samples created this way against the ones that used the ReDO segmentation stage.

Results are presented in Tables 4.5 and 4.6. We again used a small set of real samples for each domain. Column 2 indicates the number of real samples used. The following columns present the results achieved without pretraining initialization (column 3), as well as with the adoption of pretraining initialization using samples that were synthesized both with this naive pasting (column 4) and with the ReDO segmentation (column 5).

As we can see, using unsegmented classification samples already causes a significant improvement, outperforming the non pretrained models in all scenarios, and managing to produce surprisingly good results. But in all cases, they either match or under-perform the models that were pretrained on samples synthesized with segmentation, with the largest gap being of around 11% for the Birds class using the MobileNet backbone. We also note that the gap varied considerably across the two architectures (see again the birds rows). This might indicate that MobileNet (Howard *et al.*, 2017) based detectors are not robust to certain types of annotation noise, and may be picking up unimportant details during the pretraining stage, like the fact that all bounding boxes will be perfect rectangles with sharp

	# real samples	without pretraining	pretrained w. naive pasting	pretrained w. segmentation
Faces	100 (∼7%)	58.73% ± 0.62%	75.45% ± 0.41%	77.86% ± 0.46%
Birds	100 (∼1%)	44.30% ± 2.08%	70.58% ± 0.83%	81.68% ± 0.99%
Cars	100 (∼1%)	52.06% ± 1.43%	96.91% ± 1.39%	99.05% ± 0.10%

Table 4.5: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with 100 real samples, without pretraining vs pretrained with samples synthesized through naive pasting vs pretrained with samples synthesized using unsupervised segmentation. Each value is the average with standard deviation taken from three independent runs with the same training configurations.*

	# real samples	without pretraining	pretrained w. naive pasting	pretrained w. segmentation
Faces	100 (∼7%)	68.15% ± 0.72%	76.49% ± 0.18%	79.14% ± 0.25%
Birds	100 (∼1%)	76.35% ± 0.78%	81.41% ± 0.16%	83.30% ± 1.74%
Cars	100 (∼1%)	94.92% ± 1.26%	99.06% ± 0.10%	99.36% ± 0.07%

Table 4.6: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with 100 real samples, without pretraining vs pretrained with samples synthesized through naive pasting vs pretrained with samples synthesized using unsupervised segmentation. Each value is the average with standard deviation taken from three independent runs with the same training configurations.*

image transitions.

Further investigation would be needed here in order to fully understand the relationship between architectural choices, format of the classification samples and object class. Regardless of this, adopting segmented samples showed the best performance across all different scenarios (even if by a small margin), and using this form of segmentation does not incur any additional supervision penalty. We speculate that, as unsupervised segmentation techniques improve, so will the advantage of using it in comparison with the trivial naive pasting approach.

4.3.4 Difference between Real and Fake Classification Images

The image gathering step in our pipeline can either be performed by sampling real classification images, or by using a generative image model previously trained on such real images. We opted for conducting all experiments so far using GAN based image generation instead of real classification images for two reasons. The major reason being the fact that using a generative model in the first stage allows us to design this infinite stream of synthesized samples, where (almost certainly) no object instance would appear more than once. This is a conceptual advantage more than a practical one in all domains considered, as current generative models also require relatively big amounts of these classification images for training. However, as generative image models become more data efficient, and approaches like

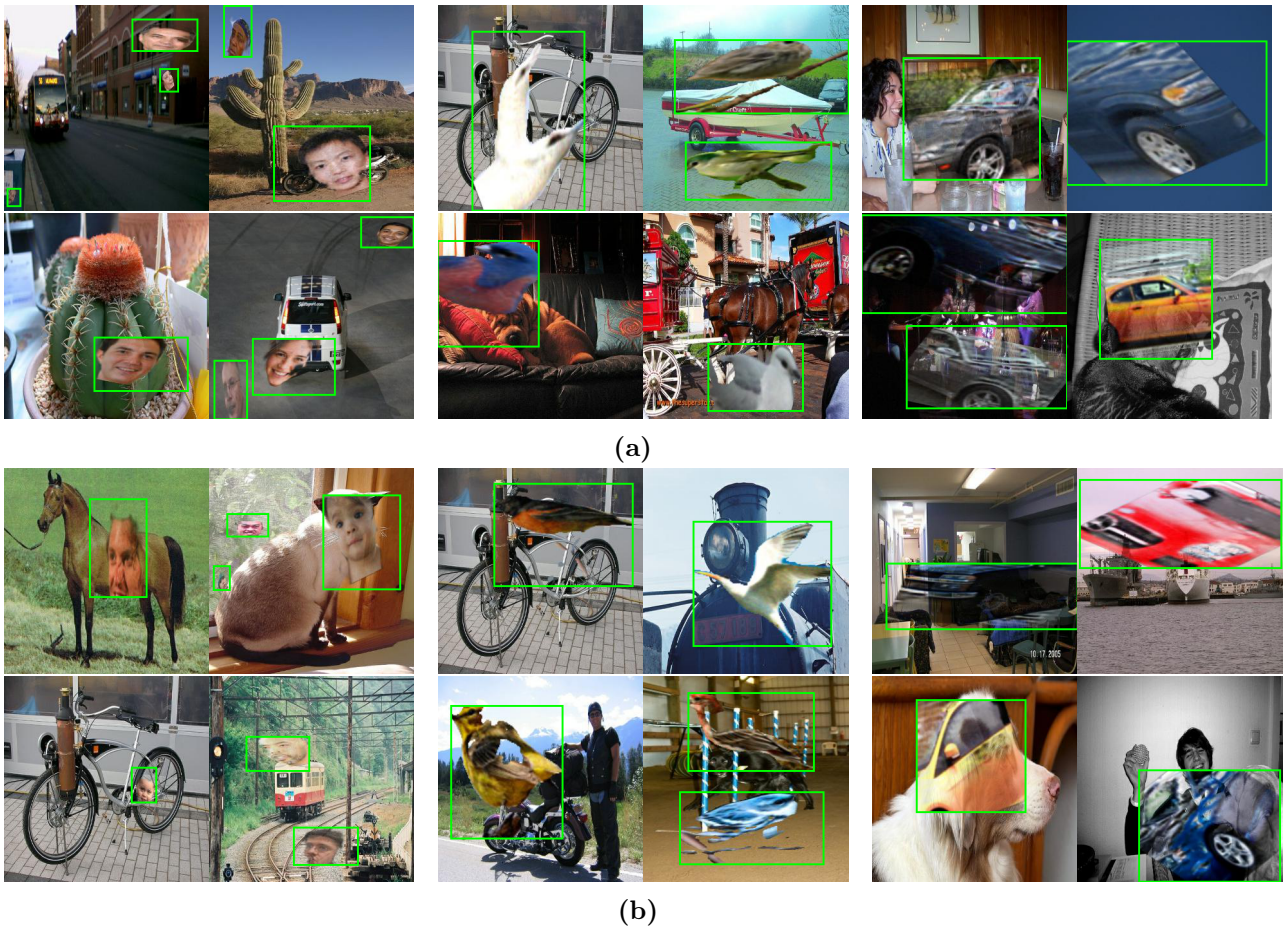


Figure 4.9: *Visual comparison between synthesized samples created from GAN generated classification images (a) and from real classification images (b).*

Unsupervised Segmentation start giving more realistic results, the advantage of this infinite stream formulation becomes more evident, and Object Detection Sample Synthesis will be capable of handling even more extreme low data situations.

Additionally, a second reason why we choose to use GAN based image generation, is because we wanted to show how sample realism is not really a significant factor for the results. Figure 4.5 already shows how the final synthesized samples are far from realistic. Yet, we demonstrate in Section 4.2 how a large enough number of these cheap samples can be considerably useful for reducing the dependency on real data.

In this section, we further demonstrate how using GAN generated classification images is equivalent to using real ones. For this, we trained a series of models using our pretraining initialization strategy, following the same configurations as before, but without the GANs for the Image Gathering stage of the synthesis pipeline. Instead, we simply sampled real images from the datasets that were used to train the respective GANs, namely the FFHQ faces (Karras *et al.*, 2019a), bounding box crops of the (non test samples) CUB-200-2011 dataset for birds (Wah *et al.*, 2011), and LSUN Cars (Yu *et al.*, 2015). Figure 4.9 compares instances of synthesized samples created from real classification images against the ones that used GAN generated classification images.

	# real samples	without pretraining	real class. images based synthesized samples	GAN generated images based synthesized samples
Faces	100 (∼7%)	58.73% ± 0.62%	77.63% ± 0.15%	77.86% ± 0.46%
Birds	100 (∼1%)	44.30% ± 2.08%	82.87% ± 1.64%	81.68% ± 0.99%
Cars	100 (∼1%)	52.06% ± 1.43%	98.91% ± 0.15%	99.05% ± 0.10%

Table 4.7: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with 100 real samples, without pretraining vs pretrained with samples synthesized based on real classification images vs pretrained with samples synthesized based on GAN generated images. Each value is the average with standard deviation taken from three independent runs with the same training configurations.*

	# real samples	without pretraining	real class. images based synthesized samples	GAN generated images based synthesized samples
Faces	100 (∼7%)	68.15% ± 0.72%	78.67% ± 0.23%	79.14% ± 0.25%
Birds	100 (∼1%)	76.35% ± 0.78%	84.87% ± 0.54%	83.30% ± 1.74%
Cars	100 (∼1%)	94.92% ± 1.26%	99.22% ± 0.14%	99.36% ± 0.07%

Table 4.8: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with 100 real samples, without pretraining vs pretrained with samples synthesized based on real classification images vs pretrained with samples synthesized based on GAN generated images. Each value is the average with standard deviation taken from three independent runs with the same training configurations.*

The results are presented in Tables 4.7 and 4.8. Column 2 indicates the number of real samples used. The following columns present the results achieved without pretraining initialization (column 3), as well as with the adoption of pretraining initialization using samples that were synthesized based on real classification images (column 4) and based on GAN generated images (column 5). As claimed, there is no apparent loss of performance by using the GAN based infinite stream approach instead of using real classification samples.

4.4 Results on the WIDER Faces

As a final experiment, we evaluated our pretraining initialization strategy on the WIDER dataset (Yang *et al.*, 2016), a state-of-the-art benchmark for face detection. This is a much larger and challenging dataset than the older FDDB (Jain and Learned-Miller, 2010), with 32,203 images and 393,703 face boxes with a much higher degree of variability. Experiments on the WIDER dataset (Deng *et al.*, 2019) usually report results on three difficulty partitions: easy, medium, and hard. These are established based on the detection rate of the EdgeBox method (Zitnick and Dollár, 2014).

We conducted this experiment using a RetinaFace model (Deng *et al.*, 2019), a detection architecture designed specifically for faces. It generates three outputs: (1) the traditional detection information (class + bounding box); (2) a key point estimation output, which is trained on facial landmarks the authors labeled on the WIDER dataset; and (3) a mesh decoder output, which is trained to produce 3D facial information in a self-supervised way. For further details about the architecture, please refer to Deng *et al.* (2019). When this experiment was conducted, the RetinaFace was the state-of-the-art architecture for the hard partition of the WIDER benchmark [6].

We initialized a RetinaFace model with a MobileNet backbone (Howard *et al.*, 2017) by pretraining it on our infinite stream of synthesized face samples, while ignoring the facial landmarks output. Next, we finetuned the model on the WIDER dataset. Due to computational constraints, we only trained for half the number of epochs (125 instead of 250) and half the image size (320 instead of 640). We considered different numbers of real samples. Results are shown in Table 4.9. Column 1 indicates the number of real WIDER samples used. The last three columns correspond to the results achieved on each of the three difficulty partitions of the WIDER dataset.

We can observe a significant improvement in the results when very few real images are used. With 100 images for instance, our pretraining initialization strategy achieves an increase of around 10% over the non initialized models, across all three difficulty partitions. This presents further evidence that synthesized detection samples can be used to improve data efficiency, and therefore improve the results on low data scenarios. However, the gap quickly diminishes as more real samples are used. This might be evidence that initializing

[6] paperswithcode.com/sota/face-detection-on-wider-face-hard

		Easy	Medium	Hard
50 real samples	w/o pretrain	40.76%	38.35%	31.22%
	w/ pretrain	49.71%	44.39%	37.00%
100 real samples	w/o pretrain	45.5%	42.57%	33.36%
	w/ pretrain	59.34%	56.40%	43.19%
250 real samples	w/o pretrain	58.65%	53.99%	45.39%
	w/ pretrain	63.21%	58.80%	49.41%
500 real samples	w/o pretrain	67.25%	64.25%	52.75%
	w/ pretrain	66.54%	64.08%	53.09%
1000 real samples	w/o pretrain	70.02%	67.30%	56.88%
	w/ pretrain	72.65%	69.24%	58.35%
all real samples	w/o pretrain	83.14%	79.36%	69.65%
	w/ pretrain	82.15%	77.78%	67.26%

Table 4.9: *Validation performance on the WIDER dataset (Yang et al., 2016) of RetinaFace (Deng et al., 2019) models, using different amounts of samples. We used the Pytorch (Paszke et al., 2019) implementation provided by* `github.com/biubug6/Pytorch_Retinaface`.

the model on a related but different data distribution of the same task could potentially harm the results.

Additionally, the gap on the low data cases tended to be slightly larger for the easy and medium partitions. We speculate this might be the case due to the images in the harder partition being considerably more different of the synthesized samples than the images on the easy and medium partitions.

Nonetheless, further investigation is required to fully understand the limitations of sample synthesis in situations where real data is already abundant, or when real data differs significantly from synthesized data.

Chapter 5

Conclusion

The overall goal of this project was to reduce the dependency of modern Object Detection techniques on expensive supervision. In particular, we tried to tackle this issue from the data perspective, by investigating whether it is possible, using existing techniques, to synthesize artificial labeled detection samples, and to understand what would be the best way of using such samples.

Overall, we believe Chapters 3 and 4 provided answers to the three questions posed in the Introduction (Section 1.1): **(1)** Can we use Sample Synthesis to achieve more data efficient Object Detection? **(2)** How to design a synthesis pipeline that does not depend on bounding box labels or other forms of expensive supervision? and **(3)** What is the best way to incorporate these synthesized samples into the training of modern detection architectures?

As also claimed in the Introduction (Section 1.1), we summarize our contributions as follows:

- We demonstrated how Sample Synthesis is a viable option to improve data efficiency for modern Object Detection models.
- We showed that, with current generative techniques, it is already possible to design a simple synthesis pipeline capable of producing an infinite stream of synthesized samples paired with bounding box labels. Moreover, all stages of this synthesis pipeline can be fully trained using only classification images.
- We proposed using this infinite stream of samples as a pretraining initialization strategy, and experimentally demonstrated how it allows existing detection models to achieve comparable results using only a fraction of the original real labeled detection data. As the whole pipeline needs only classification level images, which are considerably cheaper and more readily available than the more expensive detection images with bounding box labels, this approach effectively allows us to take advantage of a much bigger amount of data in order to reduce the need for costly supervision.

This project allowed us to identify a promising option to improve data efficiency on Object Detection, but it also made us identify several existing limitations of existing techniques

which, once overcame, would unlock the possibility of applying Sample Synthesis for more challenging detection domains.

In our experiences and investigations, we concluded that existing Generative Image Models have great potential for tasks that require synthesizing additional image samples, but they are in general still limited to generating simple classification style images. In this regard, the direction explored by the "GAN Dissection" technique (Bau *et al.*, 2019) show great potential to allow direct control over the output of GAN generators.

We also identified Unsupervised Segmentation to be the currently easiest approach to synthesize detection samples. The ReDO method (Chen *et al.*, 2019) is a recent technique that achieves impressive results on a small set of real world objects, but other approaches like "Copy-Pasting" GANs (Arandjelović and Zisserman, 2019) could also be used once their applicability extends into more realistic scenarios.

Finally, another approach that could be used for sample synthesis instead of Unsupervised Segmentation is the one taken by the LR-GAN method (Yang *et al.*, 2017), which generates images already paired with object masks.

5.1 Future Work

As an initial extension of this work, by making all stages on the proposed synthesis pipeline capable of handling more object classes, it could be scaled up to synthesize samples containing more diverse object types, or even containing several object types at the same time.

Next, by knowing that Sample Synthesis can improve the results on Object Detection problems, we open the possibility for exploring how to make the synthesis process more efficient and useful. The bottleneck limitation in the current formulation, and also the most obvious point of improvement, is the Unsupervised Segmentation stage. Future improvements on existing techniques or even novel approaches can be immediately transferred into our synthesis pipeline. Additionally, we could also directly benefit from advances on computationally efficient and less data dependent Generative Image Models.

Finally, developments on more flexible Generative Model techniques, similar to the already existing GAN Dissection (Bau *et al.*, 2019) and the LR-GAN (Yang *et al.*, 2017), could allow us to "compress" the synthesis pipeline into fewer stages, and even synthesize objects within scenes taking the surrounding context into consideration.

Appendix A

Mean Average Precision for Object Detection

As it is common practice on the Object Detection literature, all evaluations performed on this project have results reported in **Average Precision** (**AP**), in the same way as is done since the 2010 edition of popular Pascal VOC detection track (Everingham *et al.*, 2010).

In the context of Object Detection, the concepts of Accuracy, Recall, and *true/false positives/negatives* are not so intuitive as in classification problems. Therefore, we provide this Appendix to describe in further details how the Object Detection (mean) Average Precision metric is computed.

A.1 Intersection over Union (IoU)

In order to build a Precision-Recall curve, we first need to define what constitute *true/false positives/negatives* in the context of Object Detection. For this, we use the IoU metric, that scores the degree of similarity between two rectangular bounding boxes. IoU consists of the ratio between the common area between the two boxes (intersection) and the total area covered by the two boxes together (union). Figure A.1 illustrates this operation.

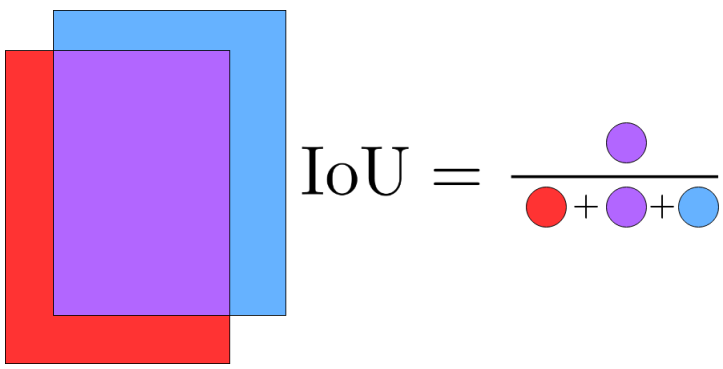


Figure A.1: *Illustrated Intersection over Union (IoU) operation.*

On the definition followed by the Pascal VOC competition (Everingham *et al.*, 2015),

two boxes correspond to the same object if they have IoU > 0.5 (Everingham *et al.*, 2010).

A.2 Recall vs Precision Curve

In order to categorize a set of detections into *true positives* and *false positives*, we check the similarity between each detection box and the ground truth boxes, measured in IoU. For now, let us consider the situation where there is just one object class.

Initially, the detections are sorted in decreasing order of their confidence score, and they are considered one by one in this order. This confidence score usually corresponds to the softmax classification output in traditional detection architectures. Each detection box B_d is associated with a ground truth box B_{gt} if the following restrictions are satisfied:

1. B_{gt} is the most similar ground truth box to B_d (maximum IoU with B_d), among all ground truth boxes on the same image as B_d.
2. IOU(B_d, B_{gt}) $>$ threshold (0.5 for Pascal VOC).
3. B_{gt} still has not been associated with any other prior detection of greater confidence.

Detections that are associated with a ground truth box are potential *true positives*, while not associated detections are considered *false positives*, and not associated ground truth boxes are considered *false negatives*. Note that there is no clear correspondence for a *true negative* in Object Detection.

Still based on the previous sorted order, we vary the confidence score threshold that is used to separate positive from negative detections (not to be confused with the IoU threshold that is used to match detections to ground truth boxes) from 1 down to 0. For each threshold θ, we compute a Recall measure as the number of detections with confidence $\geq \theta$ that are associated with some ground truth box, divided by the number of ground truth boxes. We also compute a Precision measure, as the number of detections with confidence $\geq \theta$ that are associated with some ground truth box, divided by the number of detections with confidence $\geq \theta$. This way, we create a Recall $\times$ Precision curve, as exemplified in Figure A.2.

In order to reduce the influence of the oscillations in Precision, Everingham *et al.* (2012) proposed an interpolation method. It consists on simply replacing each Precision value by the maximum Precision "to the right" in the curve. This way, the Precision monotonically decreases as the Recall increases. Let $p(r)$ be the precision value at Recall r, the new interpolated Precision value is computed as follows:

$$p_{\text{interp}}(r) = \max_{r' \geq r} p(r')$$

Figure A.3 presents the interpolated version of Figure A.2.

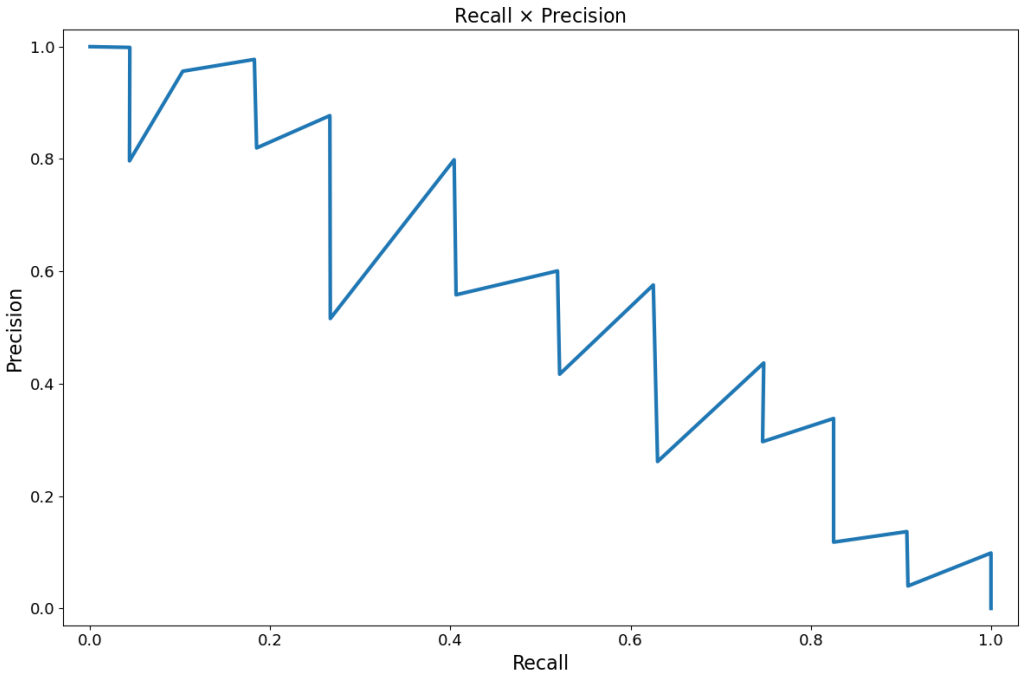


Figure A.2: *Example of a Recall × Precision curve.*

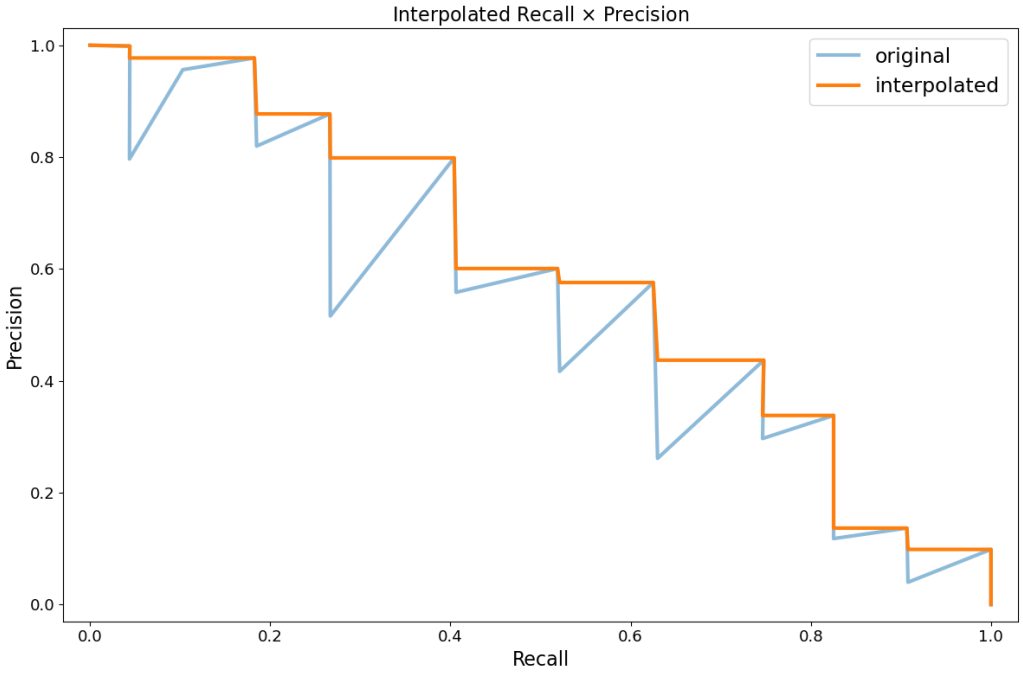


Figure A.3: *Interpolation of the Recall × Precision curve from Figure A.2.*

A.3 (mean) Average Precision

Finally, the Average Precision metric is computed as the area bellow the interpolated Recall-Precision curve. Before the 2010 edition of the Pascal VOC competition (Everingham *et al.*, 2010), the final AP was computed as the average of Precision values over the 11 equally spaced Recall points $\{0, 0.1, 0.2, ..., 1\}$.

Since the 2010 edition, the standard evaluation adopted by the competition is by computing the exact area (Everingham *et al.*, 2010). As the curve is built using a finite set of detections, every change in Precision is associated with some detection. Moreover, we have that for any detection of rank i in the previous ordering ($1 \leq i < n$), if $r_i \neq r_{i+1}$, then $p_i = p_{i+1}$. This means that Precision remains the same between consecutive Recall variations (the curve is *piecewise constant*). This property happens due to the interpolation, which causes the distinct "staircase" shape to the curve.

This way, one can calculate the exact area bellow the interpolated Recall-Precision curve with the following expression (Zeng, 2018):

$$\text{AP} = \sum_{i=1}^{n-1} (r_{r+1} - r_i) p_{\text{interp}}(r_{r+1})$$

For detection problems with multiple object classes, the main evaluation metric is **mean Average Precision** (**mAP**). It is just a simple arithmetic mean over the Average Precision of all classes.

Appendix B

Architecture Details

For the main experiments, we used two Single Shot Detection (SSD) architectures (Liu *et al.*, 2016), with MobileNet (Howard *et al.*, 2017) and ResNet50 (He *et al.*, 2016) CNN backbones, respectively. For both of them, we used the implementations of the backbone CNNs provided in the `tensorflow.keras.applications` module (Abadi *et al.*, 2015).

The SSD MobileNet architecture has four detection heads, with anchor scales set to 10%, 20%, 40%, and 80% of the image, respectively. These detection heads receive the feature maps produced by the `conv_pw_10_relu`, `conv_pw_11_relu`, `conv_pw_12_relu`, and `conv_pw_13_relu` MobileNet layers. Assuming an input shape of $300 \times 300 \times 3$ pixels, as we did in our experiments, these output heads generate detection grids with 18×18, 18×18, 9×9, and 9×9 cells, respectively. Each grid cell generates three detections, each associated with an anchor box with a different aspect ratio from $\{1/2, 1, 2\}$.

The SSD ResNet50 architecture has six detection heads, with anchor scales set to 5%, 10%, 20%, 40%, 60%, and 80% of the image, respectively. These detection heads receive the feature maps produced by the `conv4_block4_out`, `conv4_block5_out`, `conv4_block6_out`, `conv5_block1_out`, `conv5_block2_out`, and `conv5_block3_out` ResNet layers. Assuming again the input shape of $300 \times 300 \times 3$, these output heads generate detection grids with 19×19, 19×19, 19×19, 10×10, 10×10, and 10×10 cells, respectively. Again, each detection cell generates three detections, associated with anchor boxes with aspect ratios from $\{1/2, 1, 2\}$.

Figures B.1 and B.2 present diagram overviews of these architectures.

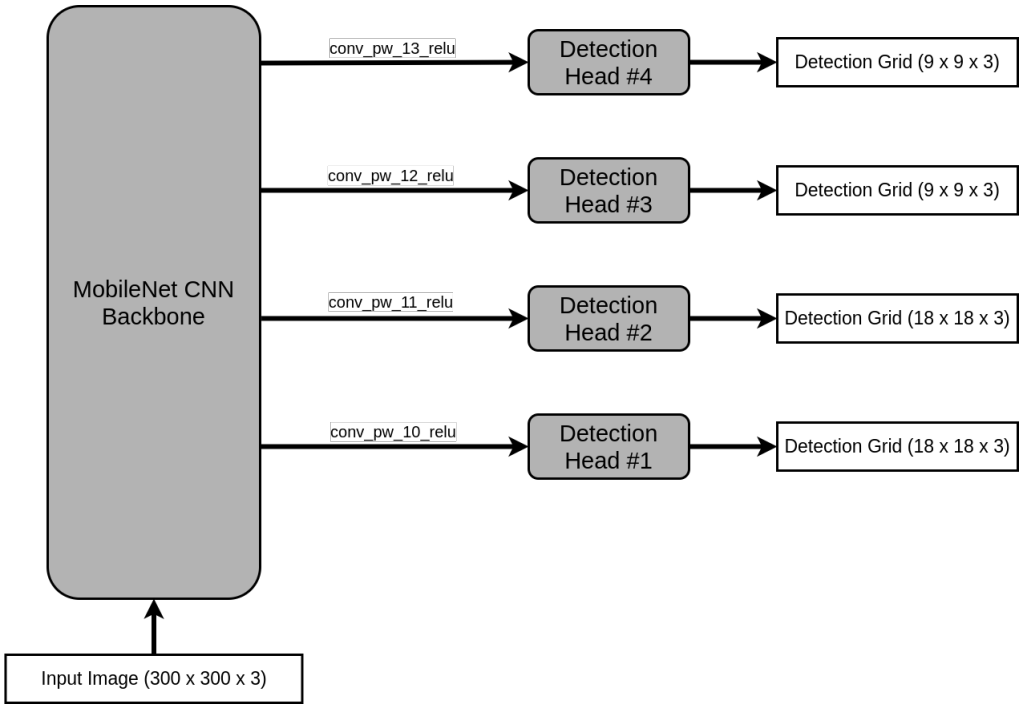


Figure B.1: *Our SSD MobileNet architecture diagram.*

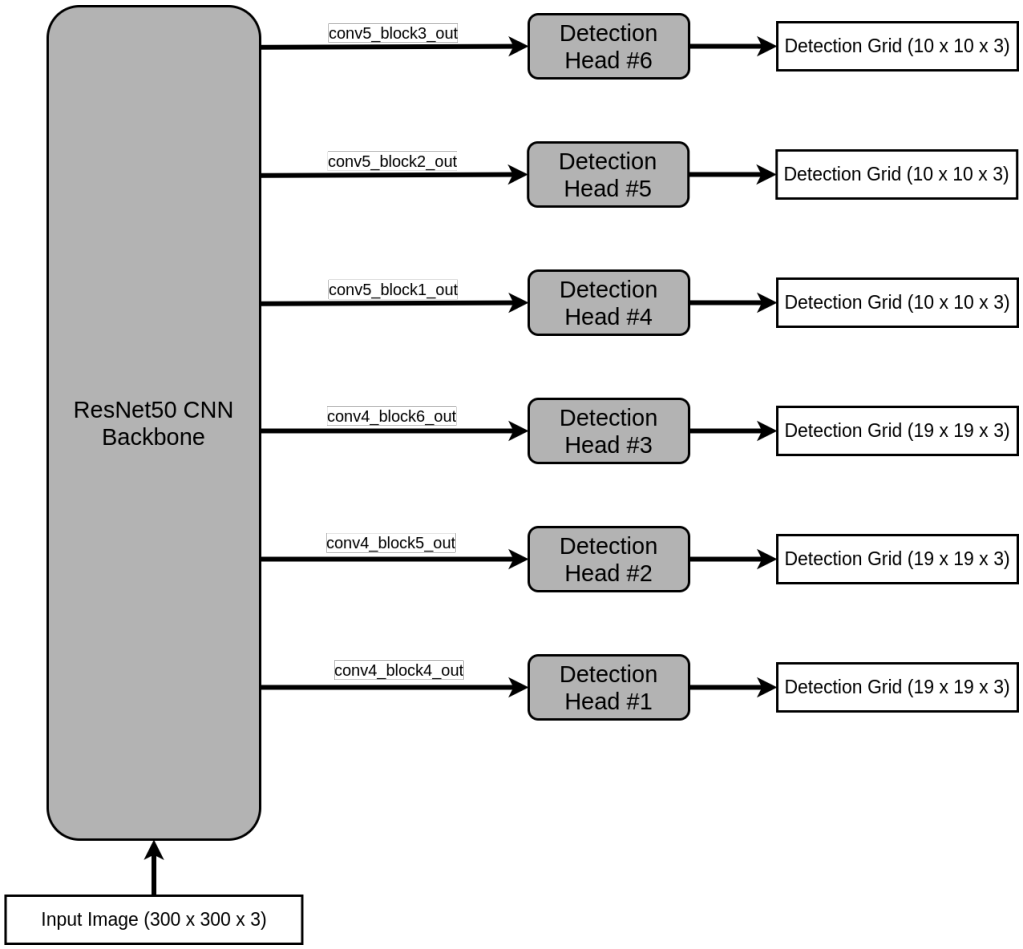


Figure B.2: *Our SSD ResNet50 architecture diagram.*

Appendix C

Detailed Results

In section 4.2, we presented numeric results for our main experiments for only a few representative numbers of real samples. In this Appendix, we provide the results for all numbers of real samples we evaluated. Note that the following Tables correspond to the graphs in Figures 4.4 and 4.7.

C.1 QR Codes

	# real samples	without pretraining	with pretraining
QR Codes	pretrain only(0%)	–	45.45% ± 0.88%
	10 (∼2%)	1.57% ± 0.30%	52.03% ± 2.91%
	50 (∼11%)	24.20% ± 0.92%	57.84% ± 0.34%
	100 (∼21%)	37.15% ± 1.48%	61.29% ± 1.45%
	150 (∼32%)	43.73% ± 1.84%	63.42% ± 1.51%
	200 (∼43%)	46.14% ± 1.40%	63.92% ± 0.82%
	250 (∼54%)	46.40% ± 0.64%	65.17% ± 1.05%
	300 (∼64%)	45.44% ± 0.59%	64.54% ± 1.48%
	350 (∼75%)	49.01% ± 0.68%	65.16% ± 1.57%
	400 (∼86%)	51.97% ± 2.58%	65.50% ± 1.02%
	467 (∼100%)	52.43% ± 3.10%	66.86% ± 1.34%

Table C.1: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# real samples	without pretraining	with pretraining
QR Codes	pretrain only(0%)	–	58.20% ± 0.89%
	10 (∼2%)	5.00% ± 1.27%	53.06% ± 1.77%
	50 (∼11%)	24.38% ± 2.87%	60.56% ± 0.81%
	100 (∼21%)	34.83% ± 2.73%	63.65% ± 0.81%
	150 (∼32%)	41.04% ± 2.66%	66.20% ± 2.19%
	200 (∼43%)	45.47% ± 0.46%	66.46% ± 0.40%
	250 (∼54%)	47.67% ± 0.62%	67.89% ± 0.23%
	300 (∼64%)	47.82% ± 3.25%	67.05% ± 1.09%
	350 (∼75%)	48.61% ± 0.83%	69.10% ± 0.03%
	400 (∼86%)	51.30% ± 0.94%	70.23% ± 0.60%
	467 (∼100%)	55.50% ± 0.87%	69.29% ± 0.30%

Table C.2: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

C.2 Faces

	# real samples	without pretraining	with pretraining
Faces	pretrain only(0%)	–	56.03% ± 3.35%
	10 (∼1%)	27.73% ± 1.02%	72.21% ± 0.74%
	20 (∼1%)	41.18% ± 3.50%	73.74% ± 0.46%
	50 (∼3%)	51.19% ± 0.82%	76.56% ± 0.33%
	100 (∼7%)	58.73% ± 0.62%	77.86% ± 0.46%
	150 (∼10%)	63.32% ± 2.79%	79.39% ± 0.39%
	200 (∼14%)	65.63% ± 0.41%	80.34% ± 0.54%
	300 (∼21%)	71.62% ± 0.10%	80.88% ± 0.10%
	400 (∼28%)	72.78% ± 0.50%	82.30% ± 0.64%
	600 (∼41%)	75.16% ± 1.08%	82.82% ± 0.41%
	800 (∼55%)	76.78% ± 2.01%	83.68% ± 0.13%
	1000 (∼69%)	77.41% ± 0.64%	84.86% ± 0.90%
	1200 (∼83%)	78.61% ± 0.26%	84.31% ± 0.47%
	1449 (∼100%)	79.36% ± 1.22%	84.86% ± 0.80%

Table C.3: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# real samples	without pretraining	with pretraining
Faces	pretrain only(0%)	–	58.40% ± 0.66%
	10 (∼1%)	18.33% ± 2.01%	70.38% ± 1.53%
	20 (∼1%)	34.36% ± 2.93%	76.14% ± 0.14%
	50 (∼3%)	54.43% ± 2.45%	78.15% ± 0.30%
	100 (∼7%)	68.15% ± 0.72%	79.14% ± 0.25%
	150 (∼10%)	72.76% ± 1.08%	80.61% ± 0.28%
	200 (∼14%)	75.40% ± 0.88%	81.88% ± 0.41%
	300 (∼21%)	77.26% ± 1.73%	83.34% ± 0.38%
	400 (∼28%)	79.40% ± 0.58%	84.21% ± 0.41%
	600 (∼41%)	82.15% ± 0.56%	85.67% ± 0.11%
	800 (∼55%)	83.57% ± 0.92%	86.26% ± 0.39%
	1000 (∼69%)	84.37% ± 0.37%	86.82% ± 0.05%
	1200 (∼83%)	85.17% ± 0.54%	86.77% ± 0.67%
	1449 (∼100%)	85.90% ± 0.13%	88.06% ± 0.11%

Table C.4: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

C.3 Birds

	# real samples	without pretraining	with pretraining
Birds	pretrain only(0%)	–	25.83% ± 5.21%
	50 (∼0%)	26.69% ± 4.09%	77.95% ± 1.82%
	100 (∼1%)	44.30% ± 2.08%	81.68% ± 0.99%
	150 (∼1%)	52.82% ± 4.35%	84.21% ± 1.24%
	200 (∼2%)	53.42% ± 2.12%	87.35% ± 0.92%
	300 (∼3%)	60.29% ± 1.20%	90.46% ± 1.10%
	400 (∼4%)	67.66% ± 1.26%	91.47% ± 1.28%
	500 (∼5%)	65.05% ± 2.15%	93.49% ± 1.53%
	1000 (∼10%)	80.23% ± 2.68%	95.48% ± 0.92%
	2000 (∼19%)	90.65% ± 1.17%	96.20% ± 0.67%
	4000 (∼39%)	95.12% ± 1.06%	97.48% ± 0.97%
	8000 (∼77%)	95.35% ± 0.99%	97.56% ± 0.69%
	10345 (∼100%)	95.83% ± 1.09%	97.78% ± 0.09%

Table C.5: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# real samples	without pretraining	with pretraining
Birds	pretrain only(0%)	–	22.98% ± 5.09%
	50 (∼0%)	68.28% ± 3.34%	79.20% ± 1.43%
	100 (∼1%)	76.35% ± 0.78%	83.30% ± 1.74%
	150 (∼1%)	81.45% ± 1.54%	87.44% ± 1.18%
	200 (∼2%)	85.36% ± 2.49%	89.01% ± 1.05%
	300 (∼3%)	88.68% ± 0.46%	92.35% ± 0.43%
	400 (∼4%)	91.93% ± 0.39%	94.51% ± 0.83%
	500 (∼5%)	93.93% ± 1.10%	94.52% ± 0.58%
	1000 (∼10%)	97.40% ± 0.43%	97.43% ± 0.51%
	2000 (∼19%)	98.43% ± 0.21%	98.20% ± 0.38%
	4000 (∼39%)	98.51% ± 0.25%	98.70% ± 0.07%
	8000 (∼77%)	99.09% ± 0.35%	99.06% ± 0.24%
	10345 (∼100%)	98.67% ± 0.28%	99.14% ± 0.55%

Table C.6: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

C.4 Cars

	# real samples	without pretraining	with pretraining
Cars	pretrain only(0%)	–	30.51% ± 0.32%
	50 (∼1%)	43.28% ± 0.67%	98.83% ± 0.17%
	100 (∼1%)	52.06% ± 1.43%	99.05% ± 0.10%
	150 (∼2%)	55.28% ± 1.40%	99.18% ± 0.02%
	200 (∼3%)	58.71% ± 1.70%	99.34% ± 0.11%
	400 (∼6%)	66.44% ± 3.14%	99.60% ± 0.02%
	1000 (∼14%)	91.92% ± 2.36%	99.79% ± 0.05%
	2000 (∼28%)	97.54% ± 0.50%	99.79% ± 0.03%
	4000 (∼56%)	97.86% ± 0.95%	99.80% ± 0.03%
	6000 (∼84%)	98.41% ± 0.13%	99.82% ± 0.01%
	7144 (∼100%)	98.42% ± 0.28%	99.83% ± 0.01%

Table C.7: *Test set performance, in mAP@0.5, for a set of MobileNet (Howard et al., 2017) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

	# real samples	without pretraining	with pretraining
Cars	pretrain only(0%)	–	7.82% ± 0.72%
	50 (∼1%)	85.11% ± 1.98%	99.35% ± 0.02%
	100 (∼1%)	94.92% ± 1.26%	99.36% ± 0.07%
	150 (∼2%)	96.96% ± 0.31%	99.45% ± 0.05%
	200 (∼3%)	98.39% ± 0.17%	99.46% ± 0.00%
	400 (∼6%)	99.57% ± 0.06%	99.63% ± 0.05%
	1000 (∼14%)	99.82% ± 0.03%	99.71% ± 0.02%
	2000 (∼28%)	99.84% ± 0.01%	99.77% ± 0.02%
	4000 (∼56%)	99.86% ± 0.01%	99.79% ± 0.02%
	6000 (∼84%)	99.88% ± 0.00%	99.81% ± 0.01%
	7144 (∼100%)	99.87% ± 0.01%	99.82% ± 0.01%

Table C.8: *Test set performance, in mAP@0.5, for a set of ResNet50 (He et al., 2016) SSD (Liu et al., 2016) models trained with several numbers of real samples. Each value is the average with standard deviation from three independent runs with the same training configurations.*

Appendix D

ICIP Paper

Before the final conception of the ideas presented in this project, and as an initial step towards understanding the modern Object Detection literature and Deep Learning programming tools, the author engaged on another project. At some point, the results on this initial project were thought be the focus for this dissertation.

This initial project consisted of a study about Deep Learning based Object Detection architectures applied to the problem of QR Code detection on natural images, a task for which only old, pre Deep Learning approaches existed, usually trained and evaluated on very limited scenarios.

One issue we noticed with this task was the absence of a well established dataset that was challenging enough to be used as a meaningful comparison benchmark between different techniques, while also being big enough to be used for training modern Object Detection techniques. For this, one of the contributions we made was to label a dataset of images for QR Code detection, which we made publicly available [1]. This dataset was later used on part of the experiments for this project (see Section 4.1).

The final dataset consisted of 567 images with a total of 1263 codes, an average of ∼2.2 codes per image. These images were retrieved from Flickr[2]. For each code, we draw a tight bounding box around the whole visible code frame, and another box around each of its visible Finding Patterns (FIPs), the smaller squares that are present on three of the code's corners.

These labels for the FIPs ended up being a motivation for our next contribution: the design of a detection architecture that takes advantage of object sub-part annotations. It consisted of a modification of the SSD architecture (Liu *et al.*, 2016), that was capable of using bounding box labels for component parts of the main objects as auxiliary supervision. We experimentally demonstrated that this additional supervision contributed to improved results for QR Code detection in certain situations, although it is applicable to any type of object.

The combination of the labeled dataset with this architectural modification resulted on

[1] https://github.com/ImageU/QR_codes_dataset
[2] https://www.flickr.com/

a paper (Blanger and Hirata, 2019) accepted for oral presentation at the 2019 International Conference on Image Processing (ICIP), that took place on September 2019 in Taipei. For details about the dataset and the architecture, please refer to the full paper, which we are attaching in full on the next five pages.

AN EVALUATION OF DEEP LEARNING TECHNIQUES FOR QR CODE DETECTION

Leonardo Blanger, Nina S. T. Hirata

University of São Paulo
Institute of Mathematics and Statistics
Rua do Matão, 1010, São Paulo, Brazil

ABSTRACT

In this work, we employ deep learning models for detecting QR Codes in natural scenes. A series of different model configurations are evaluated in terms of Average Precision, and an architecture modification that allows detection aided by object subparts annotations is proposed. This modification is implemented in our best scoring model, which is compared to a traditional technique, achieving a substantial improvement in the considered metrics. The dataset used in our evaluation, with bounding box annotations for both QR Codes and their Finder Patterns (FIPs), will be made publicly available. This dataset is significantly bigger than known available options at the moment, so we expect it to provide a common benchmark tool for QR Code detection in natural scenes.

Index Terms— QR code, deep learning, single shot detector, part-based object detection

1. INTRODUCTION

Since their release by Denso Wave Inc. in 1994 [1], QR (quick response) Codes have become increasingly present in a number of systems for a wide variety of applications. Initially designed to assist industrial processes, with the popularization of the Internet and mobile phones, they also became important for personal tasks. Nowadays, they are used to access online addresses, to make login into personal accounts, to carry any kind of textual information, among other use cases. There is a number of QR code readers/scanners that can be used for decoding by just pointing the camera towards a code and framing it. The codes can be properly decoded provided they present no geometrical distortion nor severe noise or occlusion. The raw images captured by the cameras, even when capturing is done with the explicit intention of reading a code, often present distortions such as rotation, uneven pixel intensities due to lighting conditions, perspective deformations, blur, among others. Prior to the properly said decoding step, these distortions must be corrected for a successful reading. Thus, most works related to QR code detection and recognition are concerned with the precise localization and alignment of a single QR Code on an image, so as to perform proper geometrical corrections and image enhancement prior to the decoding step [2, 3]. These works usually assume the image is well behaved in a number of useful properties (for example, presence of a single code per image, code positioned approximately at the center, no other objects present in the background, uniform light across different images), assumptions that hold when images are captured by visually healthy agents. However, visually impaired agents may even not be aware that there is a QR code in its surroundings. Therefore, detection of QR Codes in natural scenes, in the sense of the Pascal VOC [4] definition of object detection, is a key step to enable applications for use by visually impaired or robots. Once the presence of a code is detected in a scene image, then it can be adequately framed and processed for decoding as described above.

There have been very few works concerned with the detection of QR Codes in natural scenes. Existing works either use combinations of classical computer vision techniques and handcrafted algorithms [5, 6, 7, 8, 9] or very rudimentary neural network approaches, based for instance on sliding classifiers and image pyramids [10, 11]. There has been little to none work investigating the effectiveness of modern, deep learning based, object detection techniques. We review the main related works in Section 2.

The lack of public and annotated datasets is possibly one of the reasons behind the reduced number of works. A first contribution of this work is therefore the release of an annotated dataset of natural scenes, comprising images of wildly different contexts and resolutions. The images are annotated with bounding boxes for both the QR Codes and the Finding Patterns (FIPs, a squared pattern present at three corners of every code). We also establish a fixed split for training/validation/test which we used in our evaluations and that can be used in future as a reference. See Section 4.1 for the details. The second and main contribution of this work is the evaluation of a few variations of the popular Single Shot Detector architecture (SSD) on the task of QR Code detection. More specifically, we use the recent adaptation called Pyra-

Leonardo Blanger was supported by the Brazilian National Council for Scientific and Technological Development, under the process 131802/2018-6, and by the São Paulo Research Foundation, under the process 2018/00390-7. This work is supported by FAPESP (grants 2017/25835-9, 2015/22308-2). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

mid Pooling Network (PPN). Additionally, we also evaluate whether FIPs are useful or not to improve detection results. For that, we propose a modification of the SSD architecture that jointly exploits annotations of subparts (i.e., of FIPs) and of whole codes. Results are compared to the ones obtained with a method based on the Viola-Jones framework [12], proposed in [7], achieving substantial improvement. Our model is able to detect QR codes in very challenging cases. Examples of detection with our model are shown in Figure 1. The proposed detector architecture is detailed in Section 3 and results are detailed in Section 4. Our concluding remarks are presented in Section 5.

Fig. 1. Some samples from our test results.

2. RELATED WORK

In [2], the authors used a variation of the Hough Transform to detect the QR Code blocks. In [3], the authors used a contour tracking approach to locate the codes under variations of light and perspective, although their experiments showed a performance degradation when too extreme rotations were applied. Although effective, both of these methods seem to focus on images with well-behaved acquisition properties.

In [5], the authors used QR Codes to tag objects and guide a robot in an indoor environment. One part of their system required the robot to identify the QR Code location based on visual information. The way they solved it was by creating an artificial marker placed physically around the QR Code tags, containing an identifiable color (and carrying other pieces of information as well). In our work, we investigate techniques that could tackle this problem for cases where we do not have control neither over the structure or content of the codes, nor over the environment in which they are present.

A few works offer a more clear approach to natural scenes, with more than one code per image and complex backgrounds. In [8], a detection pipeline is proposed. This pipeline first preprocesses the input image converting it to grayscale, removes noise, and binarizes it, then it uses a classifier with the Local Binary Pattern (LBP) operator [13] as input, in combination with a contour approximation to propose detection regions, which are in turn refined using information about the Finder Pattern ratios of the code. In [9], the authors used a quad-tree structure to extract features based on the Histograms of Oriented Gradients [14] (HOG) over multiple image scales. In [7], the popular Viola and Jones cascade of simple classifiers [12] is used to detect FIPs. Next, triplets of detected FIPs compatible with the geometrical restrictions of the QR Code structure are considered as the final output.

Although effective, these previous works are focused only on classical techniques and hand-crafted algorithms, and there have been very few works investigating the performance of (deep) neural networks, despite their recent successes on object detection. Grósz et al. [10] trained a neural network to perform binary classification on image blocks, in order to identify which blocks are part of QR Codes. The result of this approach is an image heatmap, with clusters of high probabilities associated with the QR Code regions. Chou et al. [11] adopted a similar technique, with the additional consideration of blocks over multiple scales, considering a spatial pyramid of the image. The results are combined into the final detection through a hand-crafted pipeline of traditional image processing techniques. This last set of works, although using neural networks, do so in a local manner, through sliding window binary classifiers, image pyramids, and having the networks as a subcomponent of a more traditional pipeline of tasks. However, more recently, there has been an increase in object detection performance using end-to-end deep learning architectures [15, 16, 17, 18, 19, 20]. We also note that none of the previous works provide results in terms of (mean) Average Precision (AP), which is the most well accepted metric for the evaluation of object detection techniques nowadays [4], and also the main comparison benchmark for modern architectures. In this work, we perform an investigation of the performance of modern day deep learning object detection techniques in the task of QR Code detection, providing quantitative results in terms of AP, and releasing the labeled dataset of natural scenes used in the experiments, in order to allow future comparisons.

3. EVALUATED ARCHITECTURES

In this work, we choose to use a variation of the popular Single Shot Object Detector [17] (SSD) algorithm. We opted for the SSD due to its implementation simplicity and efficiency. In the next two sections, we present an overview of the SSD

architecture, as well as our modification to take into account object subparts annotation.

3.1. The SSD Object Detection architecture

The SSD architecture consists of a base network, usually a well known CNN architecture such as the VGG16 [21] or ResNet50 [22], optionally pre-trained for classification on the Imagenet dataset [23], maybe with a few additional convolutions at the end, with a number of its layers used as input to a detection head, which is responsible for predicting both classification scores and localization offsets for a fixed grid of default boxes of varying aspect ratios and scales (anchors).

Among the adaptations and improvements over the basic fixed grid approach of the original SSD paper [17], we can mention the adoption of the Focal Loss [20] instead of plain cross-entropy and the more recent Feature Pyramid Network (FPN) [19], which shares many similarities with the SSD.

For our experiments, we adopted the Pooling Pyramid Network (PPN) [18] version of the SSD, which consists of the basic SSD layout, with three main modifications: (1) the substitution of the convolutions between the layers used for detection by max-pooling layers; (2) parameter sharing between the detection heads; (3) adoption of the Focal Loss as in [20]. Our main motivation for the choice of the PPN version is its substantial decrease in parameter count over the original SSD version, which we believe better matches our moderate sized dataset.

3.2. Proposed Subparts Aided Object Detection architecture

QR Codes have a very rigid structure (although frequently disrupted in natural scenes). They are always square and with FIPs positioned on three of its corners. Under this perspective, we can expect that the final task of detecting the QR Codes could benefit from information regarding the position of the FIPs. More generally, the detection of a full object could benefit from cues provided by the detection of its subparts. Based on this idea, we propose a tweak to the SSD architecture in order to take into consideration annotation of object subparts during training.

The way this is implemented is by taking an intermediary layer of the base network, applying a series of max-pooling layers as in the PPN, and using the output of these layers as the input to a detection head responsible for the subparts detection. The same is done to a more advanced layer of the base network to perform the main object detection. Additionally, we concatenate the outputs of the subpart heads (both classification scores and localization offsets) to the inputs of the main heads. The motivation for this is to bring forward information about the subparts to aid the main detections. The proposed adaptation is visually described in Figure 2.

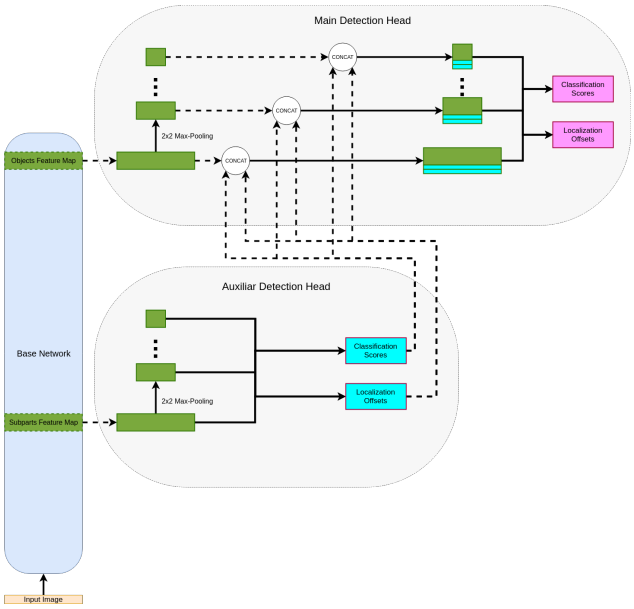


Fig. 2. Subparts Aided SSD architecture

4. EXPERIMENTAL RESULTS

4.1. Dataset

The dataset used in the experiments comprises a total of 767 images of varying sizes collected from the Internet, containing 1263 QR Codes. We have labeled both the visible regions of the QR Codes, as well as the FIPs. Annotations are available in terms of bounding boxes coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$. The dataset will be publicly available[1], with the "official" split of training/validation/test of 567/100/100 images that we have used in our tests.

4.2. Performance evaluation

In order to evaluate the performance of the mentioned deep learning models, we conducted a series of tests with varying configurations. We used three different base architectures, the VGG16 [21], used by Liu et al. in the original SSD implementation [17], the MobileNet [24], used by Jin et al. [18] in the PPN adaptation, and the popular ResNet50 [22].

For each of the base networks, we trained the corresponding SSD-PPN detector in three different scenarios:

1. **QR Codes only.** This is the standard application of our object detectors over the QR Code annotations, ignoring the FIPs. The goal of this scenario is to evaluate the more raw performance of deep learning object detection for QR Codes.

2. **QR Codes + FIPs.** Same as the previous one, but considering the FIPs as another object class, with the same

[1] https://imageu.github.io/data+code/

loss weight as the QR Codes. The goal now is to evaluate how subpart annotations could potentially contribute to the detection performance, without any architectural change.

3. **QR Codes with FIPs as subparts.** In this scenario, we evaluate the detection performance of our architectural tweak (see Section 3.2).

We considered the same training parameters for all tests: 5000 gradient descent iterations, using the Adam optimizer [25] with an initial learning rate of 10^{-4}, a decay of 10^{-4} per iteration, and the β_1 and β_2 parameters at 0.9 and 0.999, respectively. At the final iterations with these parameters, all the evaluated model's loss had stabilized and were showing diminishing improvements. All images are resized to 480×480. We also used a series of augmentations comprising noise addition, geometrical transformations, and croppings. On post-processing, we ignore the detections with confidence under 0.5 and apply Non-Max-Suppression with an IOU threshold of 0.3.

At every 50 iterations, the current model is applied to the validation set, having its performance recorded in terms of AP, and the model state which scores the best in these evaluations is kept as the final model. For the scenarios 2 and 3, only the validation AP of the QR Code class is considered.

For each combination of base architecture and the above scenarios, we applied the final model obtained to the held-out test set, and the results are shown in Table 1.

	QR Codes only	QR Codes + FIPs	QR Codes + FIPs (subparts)
MobileNet	72.7%	68.7%	71.7%
VGG16	67.7%	66.0%	67.6%
ResNet50	67.1%	64.1%	**77.0%**

Table 1. Experimental results on the test set, in Average Precision.

The first thing we notice on the results is that simply adding the FIPs as another class harms the detection performance on all three architectures. We speculate that this occurs because the reduction in model capacity dedicated to the QR Code class outweighs the additional information brought by the FIPs.

Also, we can see the subparts tweak had a performance aligned with that of the normal detection for the MobileNet and VGG16 bases, with a substantial improvement for the ResNet50. The ResNet50 Subparts SSD achieved a test AP of 77%, being the best of our evaluated models.

4.3. Comparative results

None of the few related works that also use neural network techniques [10, 11] provides code or model parameters. For this reason, we are not making any comparisons with other deep learning based approaches. We are publicly releasing our code with all model weights from the previous section experiments so that future research can benefit from it.

In order to measure the performance improvement achieved in comparison to traditional techniques, we applied the FastQR method of Belussi et al. [7][2] to our test images. Their method consists of the popular Viola and Jones cascade of simple classifiers [12] to detect FIPs, followed by a search for detections satisfying the geometrical restrictions of QR Codes. As their method does not generate confidence scores for detections, there is no way to measure performance using Average Precision. Therefore, we perform the following comparisons in terms of Recall and number of false positives (FPs), which are the metrics considered in [7].

The FastQR method comes in two alternatives: a normal one, and one with fewer stages in the cascade of classifiers, which aims at a higher Recall at a price of more false positives. We evaluate both alternatives, as well as our best model, here named Subparts PPN with ResNet50, on our test split (100 images, 158 codes), and our model achieved a substantial improvement in the two metrics, as presented in Table 2.

	Recall	False Positives Count
FastQR	51,3%	186
FastQR (shallow)	55,7%	400
Subparts PPN (ResNet50 base)	**81,0%**	**66**

Table 2. Comparison of our best model against FastQR [7].

We note that Belussi et al [7] report a recall of 90.4% with 75 false positives in 135 images. Although we have no access to their images, the results in Table 2 clearly indicates the more challenging nature of our dataset.

For all experiments, we used our own implementation of SSD/PPN[3], in Keras [26].

5. CONCLUSION

We have presented an evaluation of deep learning techniques for QR Code detection in natural scenes, and proposed a novel adaptation of the SSD architecture to allow detection aided by object subparts annotations. The proposed adaptation is implemented in our best model, which is compared against a classical technique, showing substantial improvement in the considered metrics. We are also releasing an annotated dataset of QR Codes and FIPs in natural scenes, with the "official" split used in our experiments, in order to allow future research and comparisons.

[2]Avaliable at `https://sourceforge.net/projects/fastqr/`

[3]Code (with the experiments) available at `https://github.com/Leonardo-Blanger/subparts_ppn_keras`

6. REFERENCES

[1] Denso Inc., "QR code webpage," https://www.qrcode.com/en/index.html, 2019.

[2] G. Klimek and Z. Vamossy, "QR code detection using parallel lines," in *International Symposium on Computational Intelligence and Informatics (CINTI)*, 2013.

[3] H. Qi, X. Lu, and L. Lu, "A localization algorithm for distorted or rotated QR code," in *International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2014.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] Y. Xue, G. Tian, R. Li, and H. Jiang, "A new object search and recognition method based on artificial object mark in complex indoor environment," in *8th World Congress on Intelligent Control and Automation (WCICA)*, 2010.

[6] L. Belussi and N. Hirata, "Fast QR code detection in arbitrarily acquired images," in *24th SIBGRAPI Conference on Graphics, Patterns and Images*, 2011.

[7] L. F. F. Belussi and N. S. T. Hirata, "Fast component-based QR code detection in arbitrarily acquired images," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 3, 2013.

[8] L. Tong, X. Gu, and F. Dai, "QR code detection based on local features," in *International Conference on Internet Multimedia Computing and Service*, 2014, ICIMCS '14.

[9] I. Szentandrási, A. Herout, and M. Dubská, "Fast detection and recognition of QR codes in high-resolution images," in *28th Spring Conference on Computer Graphics*, 2013.

[10] T. Grósz, P. Bodnár, L. Tóth, and L. Nyúl, "QR code localization using deep neural networks," in *International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2014.

[11] T. Chou, C. Ho, and Y. Kuo, "QR code detection using convolutional neural networks," in *International Conference on Advanced Robotics and Intelligent Systems (ARIS)*. IEEE, 2015.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2001.

[13] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, 1996.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2005.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *preprint arXiv:1804.02767*, 2018.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016.

[18] P. Jin, V. Rathod, and X. Zhu, "Pooling pyramid network for object detection," *preprint arXiv:1807.03284*, 2018.

[19] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr, "Focal loss for dense object detection (best student paper award)," in *International Conference on Computer Vision (ICCV)*, 2017.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *preprint arXiv:1409.1556*, 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2016.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *preprint arXiv:1704.04861*, 2017.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *preprint arXiv:1412.6980*, 2014.

[26] François Chollet et al., "Keras," https://keras.io, 2015.

Bibliography

Abadi *et al.***(2015)** Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. Citated in page 30, 59

Alhaija *et al.***(2017)** Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *British machine vision conference*, volume 1, page 2. Citated in page 2, 13

Antoniou *et al.***(2017)** Antreas Antoniou, Amos Storkey and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*. Citated in page 2, 14, 17, 27

Arandjelović and Zisserman(2019) Relja Arandjelović and Andrew Zisserman. Object discovery with a copy-pasting gan. *arXiv preprint arXiv:1905.11369*. Citated in page 14, 23, 54

Bailo *et al.***(2019)** Oleksandr Bailo, DongShik Ham and Young Min Shin. Red blood cell image generation for data augmentation using conditional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0. Citated in page 2, 14, 27

Bau *et al.***(2019)** David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Citated in page xi, 18, 19, 54

Belussi and Hirata(2013) Luiz FF Belussi and Nina ST Hirata. Fast component-based qr code detection in arbitrarily acquired images. *Journal of mathematical imaging and vision*, 45(3):277–292. Citated in page 8

Blanger and Hirata(2019) Leonardo Blanger and Nina ST Hirata. An evaluation of deep learning techniques for qr code detection. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1625–1629. IEEE. Citated in page xi, 4, 30, 31, 70

Bowles *et al.*(**2018**) Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*. Citated in page 2, 14, 27

Brock *et al.*(**2018**) Andrew Brock, Jeff Donahue and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*. Citated in page 2, 14

Chen *et al.*(**2019**) Mickaël Chen, Thierry Artières and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *Advances in Neural Information Processing Systems*, pages 12705–12716. Citated in page xi, 3, 14, 24, 25, 26, 35, 43, 54

Cortes and Vapnik(**1995**) Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297. Citated in page 8

Cubuk *et al.*(**2019**) Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123. Citated in page 12

Dalal and Triggs(**2005**) Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society. Citated in page 8

Deng *et al.*(**2009**) J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. Citated in page xiii, 1, 8, 11, 32

Deng *et al.*(**2019**) Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*. Citated in page xiv, 50, 51

Doersch *et al.*(**2015**) Carl Doersch, Abhinav Gupta and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430. Citated in page 15

Dosovitskiy *et al.*(**2014**) Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pages 766–774. Citated in page 15

Everingham *et al.*(**2007**) M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007. Citated in page 30, 35

Everingham *et al.*(**2010**) M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010. Citated in page 6, 55, 56, 58

Everingham *et al.***(2012)** M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. Citated in page xi, xiii, 1, 7, 30, 35, 56

Everingham *et al.***(2015)** M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136. Citated in page 1, 6, 11, 55

Everingham and Winn(2011) Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit, 2011. Citated in page 31

Felzenszwalb *et al.***(2009)** Pedro F Felzenszwalb, Ross B Girshick, David McAllester and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645. Citated in page 8

Freund and Schapire(1997) Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139. Citated in page 8

Frid-Adar *et al.***(2018)** Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331. Citated in page 14, 17

Geng *et al.***(2018)** Mingyang Geng, Kele Xu, Bo Ding, Huaimin Wang and Lei Zhang. Learning data augmentation policies using augmented random search. *arXiv preprint arXiv:1811.04768*. Citated in page 12

Ghiasi *et al.***(2017)** Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*. Citated in page 12

Gidaris *et al.***(2018)** Spyros Gidaris, Praveer Singh and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR 2018*. Citated in page 15

Girshick(2015) Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. Citated in page 9, 31

Girshick *et al.***(2014)** Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587. Citated in page 1, 9, 13

Glorot and Bengio(2010) Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. Citated in page 32

Goodfellow *et al.***(2014)** Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. Citated in page 2, 14, 17, 24, 25

Gupta *et al.***(2016)** Ankush Gupta, Andrea Vedaldi and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324. Citated in page 2, 13

He *et al.***(2016)** Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. Citated in page xiii, xiv, xv, 31, 38, 41, 45, 47, 49, 59, 62, 64, 66, 68

He *et al.***(2017)** Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. Citated in page 10, 22, 23

Hinterstoisser *et al.***(2018)** Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0. Citated in page 2, 13

Hinz *et al.***(2019)** Tobias Hinz, Stefan Heinrich and Stefan Wermter. Generating multiple objects at spatially distinct locations. *arXiv preprint arXiv:1901.00686*. Citated in page 14

Hong *et al.***(2018)** Seunghoon Hong, Xinchen Yan, Thomas S Huang and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. In *Advances in Neural Information Processing Systems*, pages 2708–2718. Citated in page 14

Howard *et al.***(2019)** Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan *et al.* Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324. Citated in page 12

Howard *et al.***(2017)** Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. Citated in page xiii, xiv, xv, 12, 31, 38, 40, 44, 46, 47, 49, 50, 59, 61, 63, 65, 67

Huang *et al.***(2007)** Gary B. Huang, Manu Ramesh, Tamara Berg and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Relatório Técnico 07-49, University of Massachusetts, Amherst. Citated in page 35

Ioffe and Szegedy(2015) Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. Citated in page 12

Jackson *et al.***(2018)** Philip T Jackson, Amir Atapour-Abarghouei, Stephen Bonner, Toby Breckon and Boguslaw Obara. Style augmentation: Data augmentation via style randomization. *arXiv preprint arXiv:1809.05375*, pages 1–13. Citated in page 12

Jain and Learned-Miller(2010) Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Relatório Técnico UM-CS-2010-009, University of Massachusetts, Amherst. Citated in page xii, 34, 36, 50

Jiao *et al.***(2019)** L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868. Citated in page 1

Jung *et al.***(2020)** Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte *et al.* imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020. Citated in page 30, 35

Kanezaki(2018) Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1543–1547. IEEE. Citated in page 23

Karras *et al.***(2019a)** Tero Karras, Samuli Laine and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410. Citated in page xi, 2, 3, 14, 17, 22, 34, 35, 48

Karras *et al.***(2019b)** Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*. Citated in page 17

Karras *et al.***(2020)** Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119. Citated in page 2, 14

Kingma and Ba(2014) Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Citated in page 31

Kingma and Welling(2013) Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. Citated in page 17

Krause *et al.***(2013)** Jonathan Krause, Michael Stark, Jia Deng and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia. Citated in page xii, 35, 36

Krizhevsky *et al.***(2009)** Alex Krizhevsky, Geoffrey Hinton *et al.* Learning multiple layers of features from tiny images. Citated in page xiii, 1

Krizhevsky *et al.***(2012)** Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. Citated in page 2, 8, 12

Learned-Miller(2014) Gary B. Huang Erik Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Relatório Técnico UM-CS-2014-003, University of Massachusetts, Amherst. Citated in page 35

LeCun *et al.***(1990)** Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404. Citated in page 8

LeCun *et al.***(1998)** Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324. Citated in page 2, 12

Lemley *et al.***(2017)** Joseph Lemley, Shabab Bazrafkan and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *Ieee Access*, 5:5858–5869. Citated in page 12

Lienhart and Maydt(2002) Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–I. IEEE. Citated in page 8

Lim *et al.***(2018)** Swee Kiat Lim, Yi Loo, Ngoc-Trung Tran, Ngai-Man Cheung, Gemma Roig and Yuval Elovici. Doping: Generative data augmentation for unsupervised anomaly detection with gan. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1122–1127. IEEE. Citated in page 14, 17

Lin *et al.***(2014)** Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer. Citated in page xiii, 1, 11

Lin *et al.***(2017)** Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. Citated in page 10, 31

Liu *et al.***(2016)** Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer. Citated in page xiii, xiv, xv, 2, 9, 12, 13, 31, 40, 41, 44, 45, 47, 49, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69

Long *et al.***(2015)** Jonathan Long, Evan Shelhamer and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440. Citated in page 2, 10, 12, 13, 22, 23

Lowe(2004) David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110. Citated in page 8

Ma *et al.***(2018)** Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131. Citated in page 12

Mariani *et al.***(2018)** Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas and Cristiano Malossi. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*. Citated in page 2, 14, 17, 27

Milz *et al.***(2018)** Stefan Milz, Tobias Rudiger and Sebastian Suss. Aerial ganeration: Towards realistic data augmentation using conditional gans. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0. Citated in page 2, 14, 27

Minh *et al.***(2018)** Tran Ngoc Minh, Mathieu Sinn, Hoang Thanh Lam and Martin Wistuba. Automated image data preprocessing with deep reinforcement learning. *arXiv preprint arXiv:1806.05886*. Citated in page 12

Mitash *et al.*(**2017**) Chaitanya Mitash, Kostas E Bekris and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 545–551. IEEE. Citated in page 2, 13

Movshovitz-Attias *et al.*(**2016**) Yair Movshovitz-Attias, Takeo Kanade and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? In *European Conference on Computer Vision*, pages 202–217. Springer. Citated in page 2, 13

Noroozi and Favaro(**2016**) Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer. Citated in page 15

Park *et al.*(**2018**) Hyojin Park, Youngjoon Yoo and Nojun Kwak. Mc-gan: Multi-conditional generative adversarial network for image synthesis. In *The British MachineVision Conference (BMVC)*. Citated in page 14

Paszke *et al.*(**2019**) Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Citated in page xiv, 51

Pathak *et al.*(**2016**) Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544. Citated in page 15

Peng *et al.*(**2015**) Xingchao Peng, Baochen Sun, Karim Ali and Kate Saenko. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286. Citated in page 2, 13

Pinetz *et al.*(**2019**) Johannes Ruisz Thomas Pinetz, Johannes Ruisz and Daniel Soukup. Actual impact of gan augmentation on cnn classification performance, 2019. Citated in page 14, 17

Ratner *et al.*(**2017**) Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in neural information processing systems*, pages 3236–3246. Citated in page 12

Razavi *et al.*(**2019**) Ali Razavi, Aaron van den Oord and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847. Citated in page 14

Redmon *et al.*(**2016**) Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. Citated in page 9

Reed *et al.***(2016a)** Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt Botvinick and Nando De Freitas. Generating interpretable images with controllable structure. Citated in page 14

Reed *et al.***(2016b)** Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele and Honglak Lee. Learning what and where to draw. In *Advances in neural information processing systems*, pages 217–225. Citated in page 14

Ren *et al.***(2015)** Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99. Citated in page 9

Ronneberger *et al.***(2015)** Olaf Ronneberger, Philipp Fischer and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer. Citated in page 2, 10, 12, 22, 23

Russakovsky *et al.***(2015)** Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252. doi: 10.1007/s11263-015-0816-y. Citated in page 8

Sandfort *et al.***(2019)** Veit Sandfort, Ke Yan, Perry J Pickhardt and Ronald M Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):1–9. Citated in page 2, 14, 27

Sandler *et al.***(2018)** Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. Citated in page 12

Sermanet *et al.***(2013)** Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. Citated in page 1, 8

Shorten and Khoshgoftaar(2019) Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60. Citated in page 11, 12

Shrivastava *et al.***(2017)** Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116. Citated in page 2, 14, 27

Srivastava *et al.***(2014)** Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. Citated in page 11

Su *et al.***(2015)** Hao Su, Charles R Qi, Yangyan Li and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694. Citated in page 2, 13

Sun and Saenko(2014) Baochen Sun and Kate Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *BMVC*, volume 1, page 3. Citated in page 2, 13

Turkoglu *et al.*(2019) Mehmet Ozgur Turkoglu, William Thong, Luuk Spreeuwers and Berkay Kicanaoglu. A layer-based sequential framework for scene generation with gans. *arXiv preprint arXiv:1902.00671*. Citated in page 14

Uijlings *et al.*(2013) Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171. Citated in page 9

Van Oord *et al.*(2016) Aaron Van Oord, Nal Kalchbrenner and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. Citated in page 17

Varol *et al.*(2017) Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117. Citated in page 2, 13

Vincent *et al.*(2008) Pascal Vincent, Hugo Larochelle, Yoshua Bengio and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. Citated in page 15

Viola *et al.*(2001) Paul Viola, Michael Jones *et al.* Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1:511–518. Citated in page 8

Wah *et al.*(2011) C. Wah, S. Branson, P. Welinder, P. Perona and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Relatório Técnico CNS-TR-2011-001, California Institute of Technology. Citated in page xii, 35, 36, 48

Wang and Perez(2017) Jason Wang and Luis Perez. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, page 11. Citated in page 2, 12, 14, 17, 27

Wang *et al.*(2018) Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807. Citated in page 14

Weiss *et al.*(2016) Karl Weiss, Taghi M Khoshgoftaar and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9. Citated in page 13

Weng(2019) Lilian Weng. Self-supervised representation learning. *lilianweng.github.io/lil-log*. URL https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html. Citated in page 15

Yamaguchi *et al.*(2019) Shin'ya Yamaguchi, Sekitoshi Kanai and Takeharu Eda. Effective data augmentation with multi-domain learning gans. *arXiv preprint arXiv:1912.11597*. Citated in page 2, 14, 17, 27

Yang *et al.***(2017)** Jianwei Yang, Anitha Kannan, Dhruv Batra and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *ICLR*. Citated in page 23, 24, 25, 54

Yang *et al.***(2016)** Shuo Yang, Ping Luo, Chen Change Loy and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Citated in page xiv, 30, 50, 51

Yu *et al.***(2015)** Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*. Citated in page 35, 48

Zeng(2018) Nick Zeng. An introduction to evaluation metrics for object detection, Dec 2018. URL https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/. Last accessed on 07/23/2019. Citated in page 58

Zhang *et al.***(2016)** Richard Zhang, Phillip Isola and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer. Citated in page 15

Zhang *et al.***(2017)** Richard Zhang, Phillip Isola and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067. Citated in page 15

Zhang *et al.***(2018)** Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856. Citated in page 12

Zhang *et al.***(2019)** Xiaofeng Zhang, Zhangyang Wang, Dong Liu and Qing Ling. Dada: Deep adversarial data augmentation for extremely low data regime classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2807–2811. IEEE. Citated in page 2, 14, 17, 27

Zhao *et al.***(2019)** Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*. Citated in page 1, 6, 8, 9, 10

Zhu *et al.***(2019)** Minfeng Zhu, Pingbo Pan, Wei Chen and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5802–5810. Citated in page 35

Zhu *et al.***(2018)** Xinyue Zhu, Yifan Liu, Jiahong Li, Tao Wan and Zengchang Qin. Emotion classification with data augmentation using generative adversarial networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 349–360. Springer. Citated in page 14, 18

Zitnick and Dollár(2014) C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer. Citated in page 9, 50

Zoph *et al.***(2019)** Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*. Citated in page 12