Comparative Analysis of Image-to-Image Transformation Learning Approaches

Augusto César Monteiro Silva

Master thesis presented
to the
Institute of Mathematics and Statistics
of
University of São Paulo
to
obtain
the degree of
Master of Computer Science

Program: Computer Science

Advisor: Prof. Dr. Nina S. T. Hirata

During the development of this work the author
received financial support from CNPq and FAPESP (grants 2019/07361-5 and
2018/00477-5)

São Paulo, February/2021

Comparative Analysis of Image-to-Image Transformation Learning Approaches

Acknowledgements

First of all, I would like to thank my supervisor professor Nina S. T. Hirata, who aided me in my research path since I was an undergraduate. Without her knowledge, her advices and consistent feedback this work would not be possible. I always felt more motivated to continue and improve the research after every meeting we had.

For sure, I would not be here without the consistent support of my family throughout these years. Socorro, Osvaldo and Arthur always helped me in remaining calm during the most stressful times.

I would also like to thank my friends at IME, for the discussions, the coffee breaks and companionship during these last seven years. Amadeu Shigeo, Lucas Helfstein, Vinicius P. Duarte and many others helped me navigate through hard times, and I would not be here without their support.

This work would not be the same without my internship at Münster. The fruitful guidance and feedbacks from professor Xiaoyi Jiang were crucial to this work, and helped me in shedding a different light to my work.

I also cannot forget to mention the dearest friends I made during my stay at Münster. Living abroad would not be as rich and pleasant as it was without the companionship of Artur, Ana Alfradique, Ana L. Peixoto, Ana F. Sales, Bruna C. de Carvalho, Júlia Moura, Júlia Pinhabel, Pedro H. Pereira, Luca Henk and many others. With all the talks, laughs and friendship during those six months, I felt like I had a home away from home.

Finally, I want to thank FAPESP and CNPq, for the funding provided. This project received financial support from the FAPESP Thematic Project 2015/22308-2. Without their scholarships, I would not be able to pursue this research.

Abstract

SILVA, A. C. M. **Comparative Analysis of Image-to-Image Transformation Learning Approaches**. 2020. Master Thesis - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2020.

Manually designing an image operator that performs a specific transformation of images is a hard and time consuming task. The problem of automatically learning image operators has been researched throughout the years. Methods that tackle this problem can be roughly divided into three types: the traditional pixelwise or sliding-window approaches, the patch-to-patch approaches enabled by recent end-to-end deep learning models, and the structurally oriented approaches based on generative techniques. Each approach has its own advantages and drawbacks. The goal of this dissertation is to study the similarities and differences among these approaches, both conceptually and experimentally. In particular, we are interested in understanding how well structural information of the images such as connected thin lines are preserved. The first contribution of this work is an end-to-end method that joins the advantages of pixelwise and patch-to-patch approaches, which we call SConvNet. A second contribution is a study that shows that the skeletal similarity based metric is well suited for evaluating handwritten document binarization algorithms in a complementary way to traditional pixelwise metrics. At last, we present an experimental comparison among representative methods of the outlined three types of approaches, with respect to traditional pixelwise as well as the skeletal similarity metrics, on two image processing tasks (retinal blood vessel segmentation and handwritten document binarization). Better pixelwise metrics were achieved by patch-to-patch methods while better structural metrics were achieved by structural approaches. This is consistent with visual inspection, which shows that structural approaches better preserve the overall structure while patch-to-patch methods generate more precise contours.

Keywords: image-to-image transformations, machine learning, convolutional neural networks, image segmentation, image binarization, structure prediction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advent of technology and the popularization of digital cameras in many accessible devices, such as mobile phones and smartphones, the amount of available digital images has been constantly increasing. With that comes a great interest in processing those images in an efficient manner. For instance, one could sharpen the edges of an image to facilitate the detection of objects, remove degradation and noise from a picture, or even highlight a specific object within a set of images. Most of these image processing techniques can be seen as an image-to-image transformation, that is, an operation that transforms a given image into a version of it that corresponds to a predetermined pattern (for instance, a smoother version of the image or an image with specific parts highlighted). Many works on image-to-image transformation are found in the literature, such as colorizing a grayscale image (Zhang et al., 2016), denoising a picture (Buades et al., 2005), removing the effects of camera shake during photographs (Fergus et al., 2006)) or transforming a sketch into a realistic picture (Chen et al., 2009). Some examples are shown in Figure 1.1, from generating a photo from a sketch to segmentation of streets from an aerial image.

Although some transformations can be easily implemented or already have an established/proven method that performs them, such as transforming a colour image into a black and white one, extracting edges from an image (Canny, 1986) or enhancing the contrast of a picture (Pizer et al., 1987), in general they are not as obvious or, in some cases, a formula that achieves the desired result may even not exist.

Machine learning based approaches are commonly employed when the underlying rules of a transformation is not obvious. In this sense, the problem of learning image-to-image transformations could be formulated as follows: given a set of images and their corresponding transformation (which must be manually created, at first), estimate a function

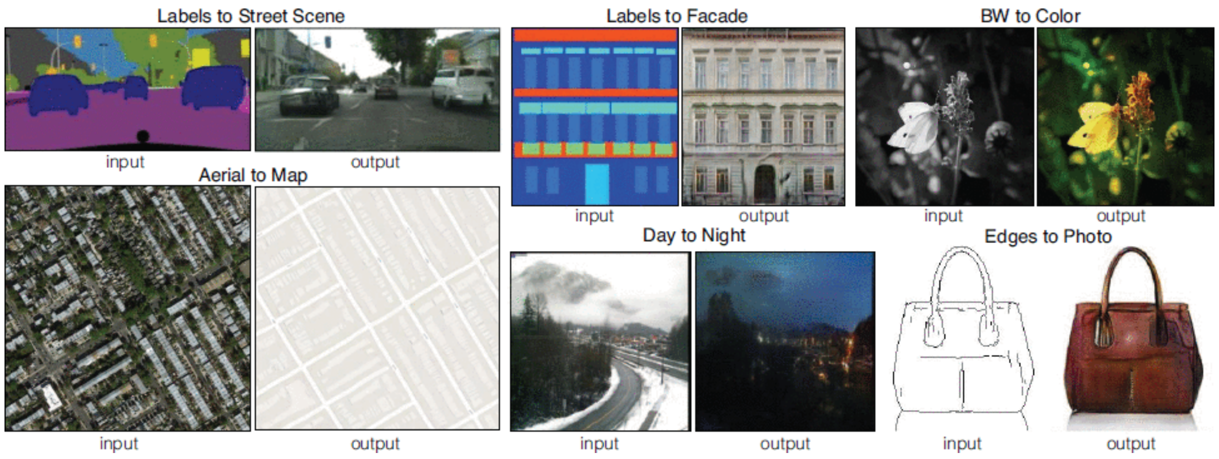


Figure 1.1: *Example of different image-to-image transformations, mapping an input image into an output one. (source: Isola et al. (2017))*

that receives an image and returns a transformed one that best approximates an expected transformation. In this work we study image-to-image transformation tasks in which the generated output image has the same size of the input image, and the pixel localization is kept. Thus, no transformations such as scaling or rotation will be considered.

Methods for automatically learning an image-to-image transformation have been extensively researched. We consider convenient to categorize them into three different approaches, depending on the techniques employed in the training phase of the algorithm, as pixelwise, patch-to-patch, or structurally oriented approaches.

Pixelwise approaches model the problem of learning an image transformation as a problem of classifying image patches, each centered on a pixel of the image. The main idea is to classify a small region of the image, and then use this classification as the value of the region's central pixel on the output image. This approach has been studied throughout the years (Barrera et al. (1997); Julca-Aguilar and Hirata (2017); Montagner et al. (2016)) and has presented good performance on locally defined transformations.

The second approach consists in training deep learning methods that receive as input an image patch and output another patch of the same size. Examples are Fully Convolutional Networks (Badrinarayanan et al. (2017); Long et al. (2015)) and UNet (Ronneberger et al. (2015)). This approach is aimed at reducing the cost of predicting each pixel individually.

The third approach consists in modeling the objects of interest in the image, and then optimizing the structure of that object in the output image. Modeling the relationship of pixels in this manner can be done explicitly (Zheng et al. (2015)) or implicitly within a deep learning architecture (Isola et al. (2017)).

1.1 Motivation

To the best of our knowledge, there is no systematic comparison between image-to-image transformation learning approaches. Thus, given a task, it is unclear which approach would be best suited for learning the transformation. Image processing tasks are diverse and often related to distinct needs. For instance, while for autonomous vehicles a real-time semantic segmentation of natural scene images is critical, for automatic disease diagnosis from medical images an accurate segmentation is much more critical than the processing speed. In contrast, for handwritten document binarization the most important aspect is the structural consistency between the strokes in the original and binarized images, rather than processing speed or pixel-level accuracy.

Some differences of the approaches can be seen in the description of the method. For instance, Fully Convolutional Networks (FCN) were motivated by the inefficient prediction of pixelwise approach, as the latter must be applied to every pixel iteratively (Long et al., 2015). FCN creates a dense prediction of the image by predicting all of the pixels at once. However, the information about the pixel location is lost during this transformation, which may lead to blob-like regions in the output image. Hence, FCN can be suited for autonomous vehicles but it is not efficient for automatic disease diagnosis. Besides, most methods optimize a pixelwise metric, trying to match every pixel of the output to a ground truth image. This may not be appropriate to some tasks, such as document binarization, in which the structural consistency is more important than an accurate segmentation. Structurally-oriented approaches were created to tackle this problem.

In order to choose the adequate method for a specific task, one must understand the unique characteristics of each. Therefore, it is important to have a systematic comparison of the approaches.

1.2 Objectives

The main objective of this work is to study and analyze approaches to image-to-image transformation learning, from the theoretical and practical point of view. We chose to follow the categorization previously described.

More specifically, the objectives of this work can be listed as follows:

- Study the three approaches (pixelwise, patch-to-patch and structurally oriented) and identify the specific advantages and drawbacks of each.

- Investigate and implement methods that could unite the advantages of each approach.
- Experimentally evaluate the performance of methods from each approach, regarding its pixelwise performance evaluation metrics and structurally-oriented metrics.

1.3 Contributions

The main contributions of this thesis are:

- Compilation of advantages and drawbacks of image-to-image transformation learning methods within the three approaches:
  - For local transformations, pixelwise methods produce well defined objects. However, time cost for training and prediction is higher.
  - Patch-to-patch methods are fast during training as well as predicting, but they lose pixel location information during the process. Output of these methods can have blob-like structures.
  - Structurally oriented approaches present better performance with respect to structural metrics than other methods maintaining better structural consistency between the ground truth and the prediction. However, these methods are harder to model and to train.
- Development of SConvNet, an image-to-image learning method that is fully convolutional and preserves pixel location, thus combining the advantages of both pixelwise and patch-to-patch approaches. SConvNet is a simpler version of fully convolutional networks, and a comparison between SConvNet and other methods is presented in Chapter 5.
- Investigation about the use of Skeletal Similarity metrics for the evaluation of document binarization algorithms. These metrics better capture the structure consistency between two images, thus, they can be used to evaluate and compare different binarization algorithms, in a complementary way to traditional pixelwise metrics. This investigation resulted in a paper presented at the *International Conference on Frontiers in Handwriting Recognition 2020* (ICFHR 2020) (Silva et al., 2020).
- An extensive experimental comparison between the studied methods from the three approaches, applied on retinal blood vessel segmentation and handwritten binarization, including:

- Performance according to common pixelwise metrics
- Performance according to skeletal similarity metrics

1.4 Thesis Structure

The structure of this thesis is as follows: Chapter 2 presents a brief review on the concepts of the three approaches, and describes example methods from each of the three approaches. It also describes SConvNet, a simple method that combines the advantages of pixelwise and patch-to-patch approaches. Chapter 3 presents a review of metrics used for evaluating image-to-image transformation algorithms, both pixelwise Sensitivity, Specificity and Accuracy, as well as structurally oriented Skeletal Similarity. Chapter 4 presents the investigation about using Skeletal Similarity metrics, which were designed for evaluation of vessel segmentation, to evaluate document binarization algorithms. Chapter 5 presents an experimental comparison of some of the methods described in Chapter 2 according to the evaluation metrics described in Chapter 3. Chapter 6 reports the conclusions of this work.

Chapter 2

Image-to-Image Transformation Learning Approaches

Methods for image-to-image transformation learning can be divided into three approaches based on how the image operator is modeled: a pixelwise approach (learning the relationship between an image patch and the output value of its central pixel), a patch-to-patch approach (learning the transformation between a patch and the output values of the pixels in the entire patch) or a structurally oriented approach (learning the structure of the objects of interest in the image by modeling the relationship between the pixels in the output image). This chapter describes in more detail each of these approaches, highlighting differences, advantages and drawbacks, while also describing specific methods of each. We chose to study methods that highlighted the differences of the approaches and that had already shown good performance in previous works. Section 2.1 explains the pixelwise approach and describes the family of W-Operators, the pixelwise modeling used in this work. Section 2.2 explains the concepts of patch-to-patch approach, how it differs from the pixelwise case, and details Fully Convolutional Network and UNet, two patch-to-patch methods. Section 2.3 describes the concept of structurally oriented approaches and details two methods that model the structure of the transformed image. Section 2.4 describes SConvNet, a simple architecture that unites the advantages of pixelwise and patch-to-patch approaches.

2.1 Pixelwise approach

Traditional image processing algorithms are based on pixelwise processing. Typically, a function that receives image patches as input and returns scalar values is applied pixel-by-pixel, with each image patch being centered on one pixel and the output of the function being assigned to the same pixel in the output image. This process is often called *sliding window* approach, and the window usually is a rectangular shaped parameter that defines the patch size. Hence, a natural machine learning approach to design image transformations would be to learn these local functions from training images.

A large body of studies on image-to-image transformation learning tackles the design problem this way (Julca-Aguilar and Hirata (2017); Montagner et al. (2017)), and most of them consider a family of operators that are translation invariant and locally defined, called W-Operators. Next, we formally define W-Operators and describe how W-Operator learning works.

2.1.1 W-Operators

To keep the mathematical formulation simple, we restrict ourselves to grayscale images. For convenience, we assume images are defined on the grid $\mathbb{E} = \mathbb{Z}^2$ instead of on a limited support region. Any grayscale image can be modeled as a function $f : \mathbb{E} \to K$ where $K = \{0, 1, ..., k-1\}$ is the set of k gray levels. For every pixel p, $f(p)$ is the intensity value of the pixel in image f. The set of all grayscale images with k gray levels is denoted as $K^{\mathbb{E}} = \{f | f : \mathbb{E} \to K\}$.

An image operator is a function $\Psi : K^{\mathbb{E}} \to K^{\mathbb{E}}$ that takes a grayscale image f as an input and returns a transformed one $\Psi(f)$. The intensity value of pixel p in the transformed image $\Psi(f)$ is denoted $[\Psi(f)](p)$. Note that every image g with l gray levels such that $l \leq k$ is contained in $K^{\mathbb{E}}$, thus, image transformations that produce binary images, such as binarization or segmentation are also included in this family of image operators.

To define a W-Operator, we first need to define the properties of translation invariance and local definition. Denote f_q as the translation of image f by vector $q \in \mathbb{E}$, i.e., $f_q(p) = f(p - q) \quad \forall p \in \mathbb{E}$. An example of an image f and the respective translation by q can be seen in Figure 2.1.

An image operator is translation invariant if, for any image and any translation, the output image obtained by applying the operator to the translated image is equivalent to translating the output image obtained by applying the operator on the original image. In other words, an image operator Ψ is considered to be translation invariant if for any

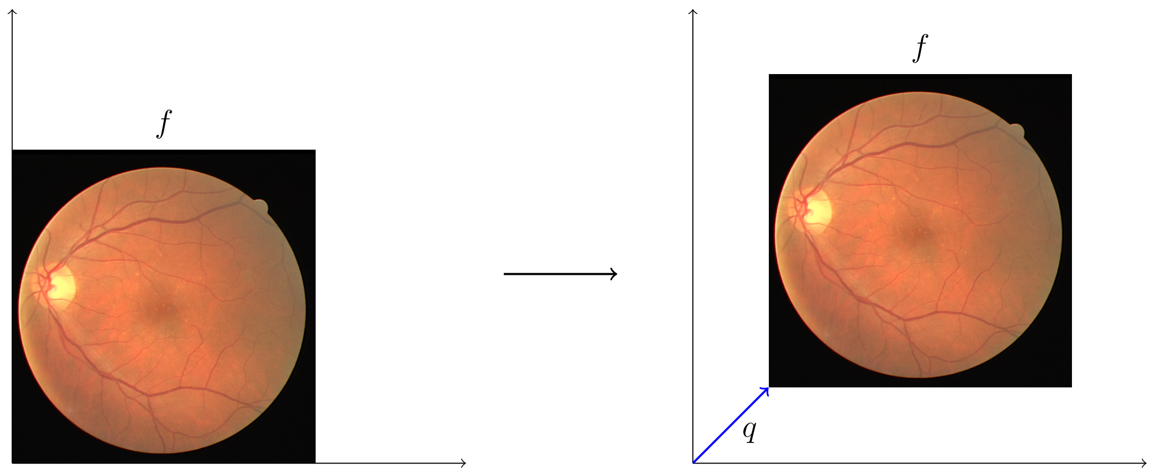


Figure 2.1: *Example of image f, and image f_q (f translated by vector q)*

$f \in K^{\mathbb{E}}$ and $q \in \mathbb{E}$, the following holds:

$$\Psi(f_q) = [\Psi(f)]_q \tag{2.1}$$

An image operator is locally defined if there exists, for any pixel p, a neighborhood region such that when the operator is applied to the image restricted to this region, the output is the same as when the operator is applied to the image restricted to any larger region. Formally, an image operator is locally defined if there exists a finite window (set) $W \subseteq \mathbb{E}$ such that, for any $p \in \mathbb{E}$, $f \in K^{\mathbb{E}}$ and $W' \supseteq W$ we have

$$[\Psi(f)](p) = [\Psi(f|_{W'_p})](p) \tag{2.2}$$

Here $f|_{W'_p}$ denotes the image f restricted to W'_p, that is, $f|_{W'_p}(q) = f(q)$ if $q \in W'_p$ and $f|_{W'_p}(q) = 0$ if $q \notin W'_p$.

An operator that satisfies both properties, translation invariance and local definition, is called W-operator and is uniquely characterized by a function $\psi : W^K \to K$ (Heijmans (1994); Montagner et al. (2016)) that, for any image $f \in K^{\mathbb{E}}$ and $p \in \mathbb{E}$, satisfies:

$$[\Psi(f)](p) = \psi(f_{-p}|_W) \tag{2.3}$$

The characteristic function ψ is also called local function.

2.1.2 W-Operator Learning

As any W-operator Ψ can be uniquely characterized by a function ψ, learning such operator is equivalent to learning their characteristic function. Moreover, learning such function can be cast as a classification problem, where the input space consists of every possible image patch of size equal to the size of W and the classes are the possible pixel intensity values in the output image. For instance, in the image binarization task, the problem of designing an image operator Ψ would be modeled as a binary classification problem, where each pixel needs to be classified as background or foreground. The input for the classifier would be the image patches centered on each of the pixels, and the classifier output would be assigned to the respective pixels in the output image.

To train the classifiers, a set of input-output pairs (f, g) of training images is required. The training process is as follows. We define a window W and slide it over each input image f in the training image set, extracting the patch under the window at each pixel p (i.e., $f_{-p}|_W$) and also the value of the same pixel in the respective output image g, i.e., $g(p)$. Extracted patches and corresponding output pixel values form the training set. After the classifier is trained, it can be applied in a pixelwise fashion as described before, filling the output image one pixel at a time. A diagram of training and prediction steps of this method can be seen in Figure 2.2. In the diagram, f represents the input image, g represents the ideal (target) image, $\hat{\psi}$ the estimated function, and $L(\hat{\psi}(.), g(.))$ is the loss function between an output value of the estimated $\hat{\psi}$ function and the correspondent pixel value on image g.

Methods that fall within this approach have been extensively researched throughout the years and have been successfully applied on many image processing tasks, including filtering problems (Dellamonica Jr. et al. (2007)), segmentation tasks (Calvo-Zaragoza et al. (2017); Julca-Aguilar and Hirata (2017); Montagner et al. (2017)), and biomedical imaging (Cireşan et al. (2012); Melinscak et al. (2015)). These methods use several classifier algorithms such as Support Vector Machines (SVM), Decision Trees or Convolutional Neural Networks (CNNs (Goodfellow et al., 2016)).

In Chapter 5 we experimentally evaluate the method proposed in Julca-Aguilar and Hirata (2017). This method uses a basic CNN architecture, consisting of two building blocks with convolutional, ReLU and pooling layers, followed by two fully connected layers and a final softmax layer, as a classifier.

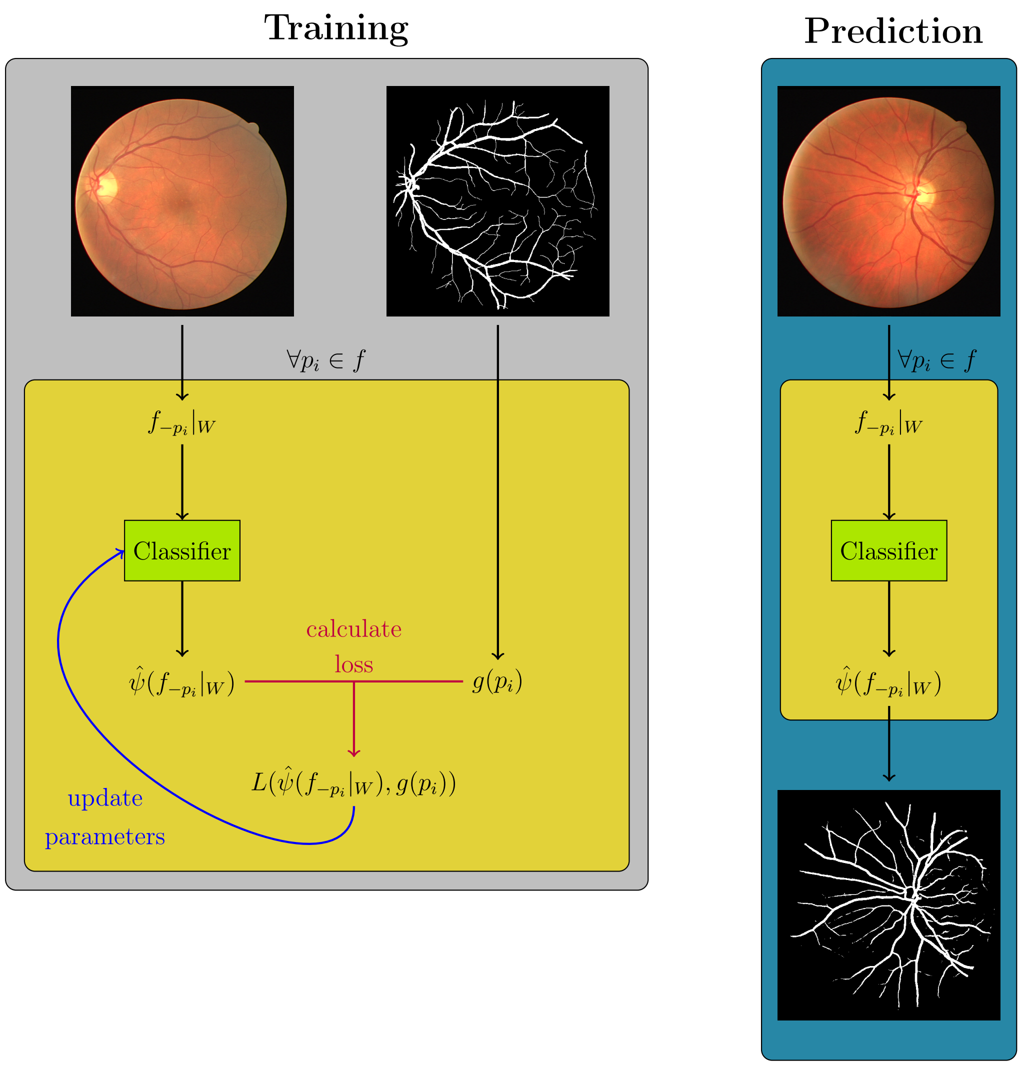
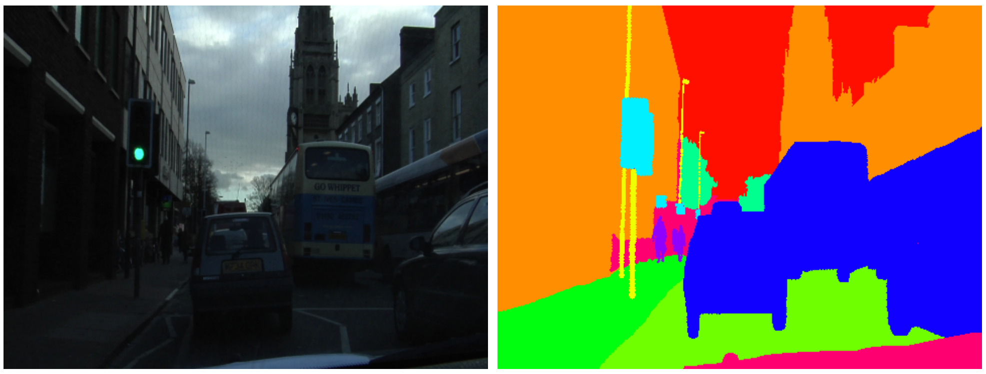


Figure 2.2: *Diagram of sliding window method for image-to-image transformation learning. The left diagram illustrates the training algorithm of sliding window, while the right diagram illustrates the prediction step. In both diagrams, the yellow block represents a loop over the pixels of the input image.*

2.2 Patch-to-Patch Approach

With the great success of deep learning techniques applied in computer vision tasks, such as Convolutional Neural Networks applied in classification problems in the works of Krizhevsky et al. (2012); Liu and Deng (2015); Szegedy et al. (2015), researches on semantic segmentation also started to use the same techniques. Semantic segmentation is a specific type of image-to-image transformation in which every pixel receives a label that identifies the object to which it belongs. An example of semantic segmentation can be seen in Figure 2.3.

Although pixel-by-pixel methods can learn a large family of image-to-image trans-



(a) *Natural scene image* **(b)** *Semantic Segmentation*

Figure 2.3: *Semantic segmentation of a natural scene image, in which each object from Figure 2.3a is represented by a color in Figure 2.3b (automobiles are dark blue, sky is red, traffic lights are light blue, street is light green, sidewalks are darker green,buildings are orange, people are purple, poles are yellow, and undefined pixels are pink colored)*

formations, time consumption of applying the learned function pixel-by-pixel to predict the image is dependent on image size, which makes prediction of large images very inefficient. Thus, to mitigate this issue, a deep learning architecture called Fully Convolutional Neural Network (FCN) was proposed in Long et al. (2015). FCN is fully convolutional (meaning that it has no fully connected layers) and thus it can process images of arbitrary size. It also outputs an image of the same size of the input image, on a single forward pass. Soon after, another model called UNet was proposed in Ronneberger et al. (2015) improving some weaknesses of FCN.

2.2.1 Fully Convolutional Networks

To the best of our knowledge, Fully Convolutional Network (FCN) (Long et al., 2015), proposed for image segmentation tasks, is the first image-to-image transformation learning method that did not rely on learning a function to be applied on each pixel of the image, but instead could be applied to the whole image at once.

This architecture was created by replacing every fully connected layers of standard convolutional neural networks used in classification tasks with convolutional layers. This can be done by translating each node of the fully connected layer to a filter of a convolutional layer. Thus, the output of these new convolutional layers are n feature maps instead of n-dimensional vectors. With this architecture, image input is not restricted to a specific size, and the output of the network is not a single value, but instead a reduced size output image or feature map.

Another detail of FCN relates to up-sampling. Convolutional Neural Networks commonly use pooling layers in order to downsample the original image. Thus, in order to return the reduced output image size to the same size as the input image, a sequence of upsampling layers is applied. For that, the original work of FCN uses *backwards convolution* (sometimes called *deconvolution*).

One of the drawbacks of FCN is the location information that is lost during pooling and subsequent upsampling layers. This causes the resulting image to appear pixelated, containing blob-like structures without the fine details in the contour of the regions. To improve on this, FCN added skip connections that combine shallow layers with the results of the upsampling. In the original work the authors made three architectures, each one adding more skip connections (FCN-32s without skip connection, FCN-16s with one skip connection and FCN-8s with two). This architecture is shown in Figure 2.4.

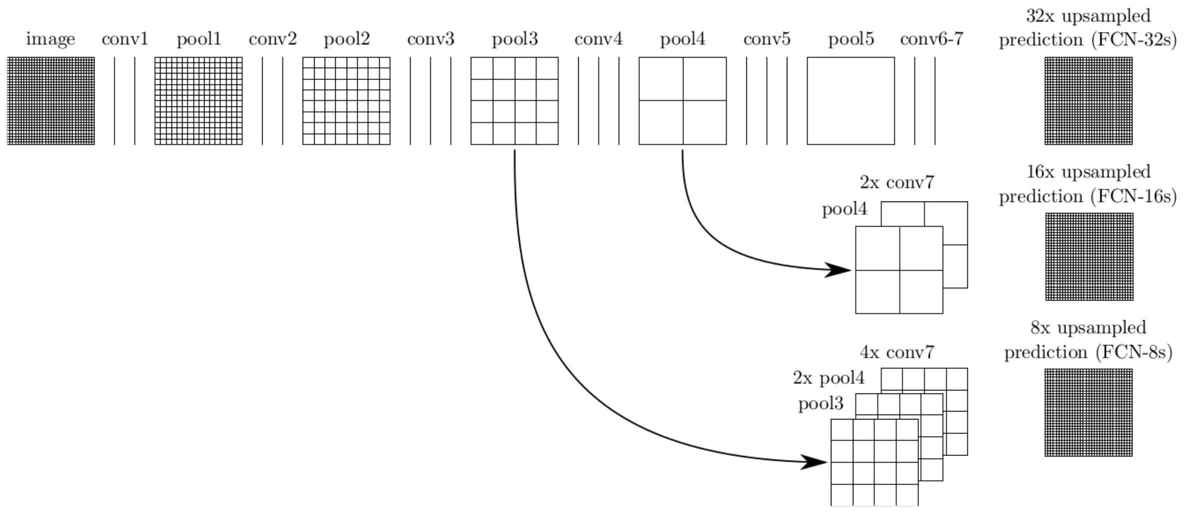


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 2.4: *FCN architecture image (source: Long et al. (2015))*

2.2.2 UNet

In order to improve on previous results achieved by FCN, a new architecture called UNet, based on an encoder-decoder structure, was proposed in Ronneberger et al. (2015). It achieved great success in semantic segmentation, and later became the standard for many applications in different fields and the base for novel architectures, such as the one proposed in Isola et al. (2017).

UNet's encoder-decoder type of architecture is divided in modules, in which each module on the encoder path have a correspondent module on the decoder path. Each encoder module consists of a sequence of convolutional layers followed by a pooling layer, and each decoder module consists of a concatenation of the output feature maps of the correspondent encoder module, a sequence of convolutions and an upsampling layer. This architecture can be drawn as a U-shaped network as shown in Figure 2.5, where the first half is the encoding part that gradually reduces the image dimension and the second part is the decoding part that gradually recovers the original image dimension. The concatenation is illustrated as the horizontal skip connections between modules. In the original work the convolutional layers with filters 3×3 are done without padding, reducing the image size by 2×2. Therefore, the output feature map of each module is not the same size as the input of the decoder module. To concatenate both, a cropped version of the output of the encoder module is used. Note that this also reduces the output segmentation map. In order to return a same sized output image, the input size must be a multiple of 2^k, where k is the amount of encoder-decoder modules, and the convolutions must be padded.

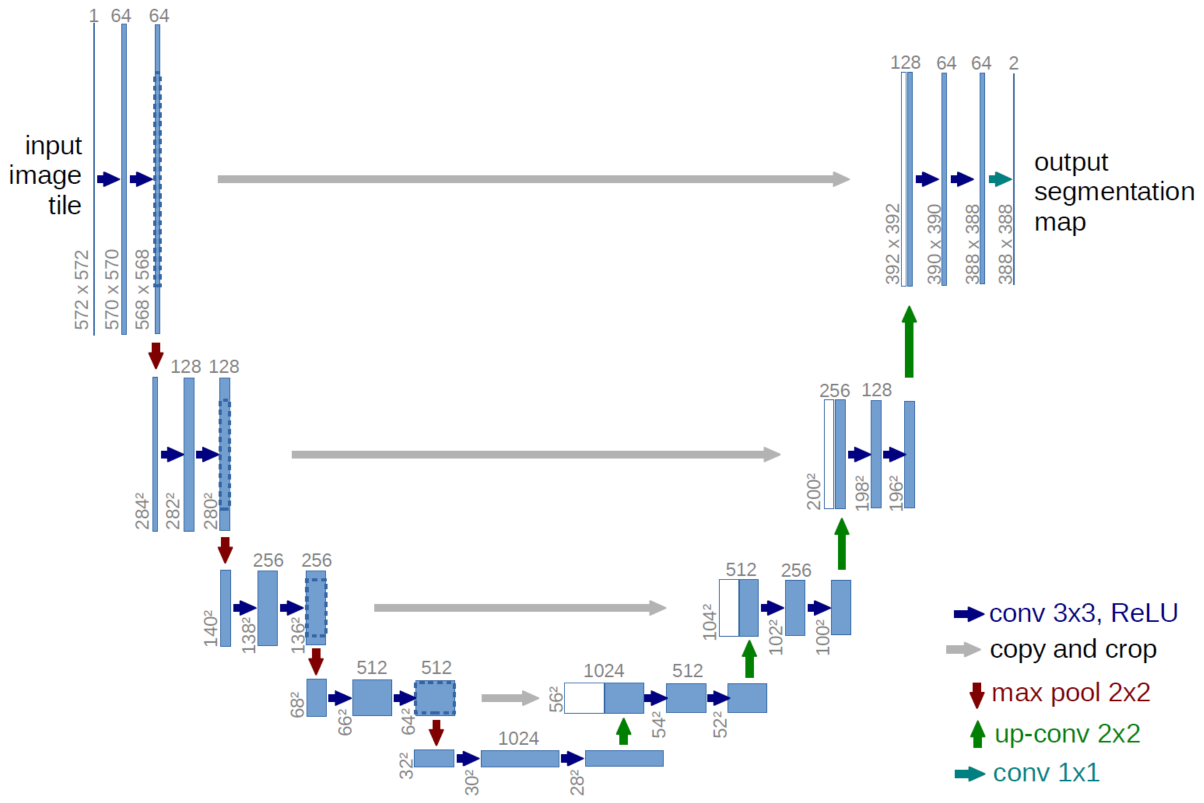


Figure 2.5: *UNet architecture as proposed in the original paper. (source: Ronneberger et al. (2015))*

2.3 Structural Approach

A third approach to the image-to-image transformation learning problem is to model the relationship between pixel output values, in order to enforce some object structures.

Modeling the relationship of output pixels could be useful to reduce output noise, such as an isolated pixel with distinct value within a region of uniform valued pixels, or for enforcing connectivity of line-like structures, or improving delineation of object contours. This is different from the previous approaches since they do not model relation among output pixels explicitly.

The structural approach is based on the idea that the output values of the pixels are not independent amongst themselves. In semantic segmentation, pixels that are near each other should have a greater probability of having the same label. For instance, if the task is to segment blood vessels from the image, it is desired that the vessel pixels of the output have a connected, thin and elongated structure.

One of the most common structural approaches for image-to-image transformation learning relies on probabilistic graphical models, which use graphs to represent the dependence between random variables. Accordingly, pixels of the image are modeled as random variables (nodes in the graph) and dependence between pixels is modeled as the edges of the graph. A well-known model is Conditional Random Fields (CRF) (Lafferty et al., 2001).

Another structural approach for image-to-image transformation learning is based on Generative Adversarial Networks (GANs), which is a framework for estimating generative models using adversarial training. Commonly used for generating images, such as realistic photographs (Brock et al. (2019)) or human faces (Karras et al. (2018)), this adversarial framework can also be used to learn image-to-image transformation, as shown in Isola et al. (2017), by conditioning the image generation with an input image. Different from CRF, output pixel relationship is indirectly modeled by the discriminator, and such relationship is learned throughout the training phase of the method.

2.3.1 Conditional Random Fields

Conditional Random Fields (also known as CRF) are a type of undirected probabilistic graph model, which represent dependence among random variables, and are commonly used on structure prediction. This model is very helpful in cases where the parts of the structures are interrelated, such as in image segmentation. The main idea of this model is that, by considering a graph $G = (V, E)$ where each vertex $u \in V$ represents a pixel in the image, pixel relationship can be modeled by the edges of such graph.

More specifically, every pixel u can be modeled by a random variable X_u that can assume a label among a set $L = \{l_1, l_2, ..., l_n\}$. Then, we can define a unary cost $\Phi_u(X_u =$

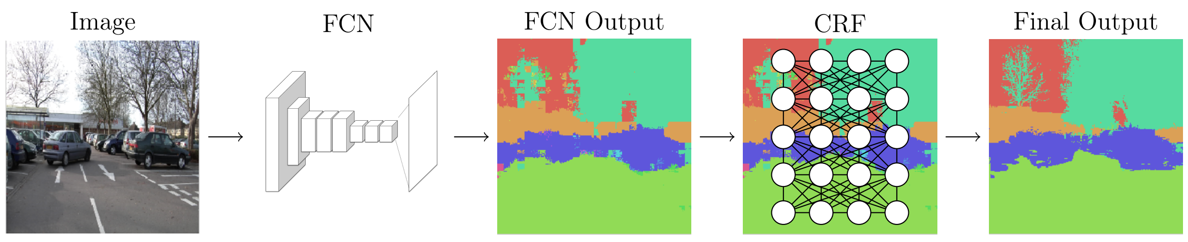


Figure 2.6: *Example of CRF post-processing method from Chen et al. (2015)*

$l_i|I)$ for each pixel, which is the cost of pixel u assuming the label l_i given image I. Although such function can model an image operator, in order to model the relation of distinct pixels, we need to define a secondary function, such as a pairwise cost. Thus, we define pairwise cost as a function $\Phi_{uv}(X_u = l_i, X_v = l_j|I)$, which represents the cost of assigning label l_i to pixel u and label l_j to pixel v given image I. More generally, we can define higher-order potentials $\Psi_{u_1,u_2,...} = (X_{u_1} = l_{i_1}, X_{u_2} = l_{i_2}, ...|I)$, which models the relationship amongst three or more variables. A weighted sum of the defined costs can be regarded as an energy of the model. Then, a minimization algorithm can be used to find a pixel-label association with minimal cost. Although this energy minimization problem is NP-Hard, as shown in Li et al. (2016), CRF algorithms commonly use approximation methods.

One of the first methods to join CRF and deep learning architectures is the work of Chen et al. (2015). In their work, Chen et al. propose the use of a Dense CRF (which models the image as a complete graph) to improve the results of semantic segmentation deep learning architectures. After training a deep learning architecture and predicting a segmentation of an image, a Dense CRF is used as a post-processing method to the segmentation, improving object contours. This Dense CRF uses only unary and pairwise costs and mean-field as the energy minimization algorithm approximation. An example of this method can be seen in Figure 2.6.

Unary cost for Chen's method is defined as the prediction from the deep learning architecture, and pairwise cost is defined as a weighted sum that depends on the distance between the pixels and the color difference between them on the original image. The pairwise cost is defined as follows:

$$\Phi_{uv} = w_1 exp\left(-\frac{||p_u - p_v||^2}{2\sigma_\alpha^2} - \frac{||I_u - I_v||^2}{2\sigma_\beta^2}\right) + w_2 exp\left(-\frac{||p_u - p_v||^2}{2\sigma_\gamma^2}\right) \tag{2.4}$$

where I is the intensity value of a pixel and p its position, $\sigma_\alpha, \sigma_\beta, \sigma_\gamma, w_1$ and w_2 are hyperparameters (σ controls the scale of each kernel and w their weight). The main idea is that the cost of assigning different labels to nearby pixels with similar intensity is higher than assigning different labels to distant pixels or with contrasting intensities. Thus, this method can better define edges and structures of the segmented objects.

Although Conditional Random Fields as post-processing improved the previous results on semantic segmentation, this method does not learn the relation between pixels, but instead assumes a previously defined relationship (close pixels with similar intensities are more likely to have the same class). This can be problematic if the classes are very unbalanced. For instance, if the desired segmentation is only one line of white pixels and the rest are black pixels, CRF techniques tend to erase the white line. To improve on this, researchers started to develop methods that include Conditional Random Fields in the neural networks, so that the CRF graph could be also learned from data during training, instead of having it defined previously.

One of the first works that developed a joint technique with CNN and CRF was proposed by Zheng et al. (2015). This method unrolls the iterations of the mean-field inference algorithm to create a Recurrent Neural Network (RNN), a type of neural network designed for processing sequential data. Each iteration of the mean-field algorithm can be broken down into common CNN operations, which can be further modeled into an RNN with the output of each iteration as the input of the next one. This RNN is then coupled with a previously trained CNN, which enables backpropagation through the mean field inference and into the CNN, jointly optimizing the parameters of both networks. This approach achieved a 2% improvement on the state-of-the-art methods at the time it was proposed.

Although achieving great performance, the probabilistic graphic model (PGM) used by Zheng et al. was outdated, and other PGMs on image segmentation had obtained better results with models that used higher order potentials, instead of only unary and pairwise costs. This was the motivation for the work of Arnab et al. (2016), in which two higher order potentials were carefully crafted and joined to the network in an end-to-end fashion, to overcome the drawbacks of previous models. One of the drawbacks was that the standard CRF model could not correct mistakes made by the unary potentials (in other words, if a CNN wrongly classified most pixels of a region, CRF would only propagate the error instead of correcting it). To improve on this, Arnab et. al proposed a potential based on object detection, that could correct wrongly classified pixels within the CRF model. The second crafted potential used superpixels to encourage consistency over similar regions of the image.

CRF models showed great performance in semantic segmentation, however most of the works rely on the idea that neighboring pixels with similar colors have the same class or that a pixel that have many neighbours of different classes is probably noise or is wrongly classified. This assumption is not true in cases where the objects of interest have thin structure and small contrast, such as retinal blood vessel segmentation, where CRF models can delete most vessel pixels, only worsening the output image. Therefore, we do not include results of the explicit structural modeling of CRF in the experimental part of this work. The next subsection describes the implicit structural modeling ability of adversarial networks, which is equivalent to learning a probabilistic graph model like the CRF.

2.3.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) is a framework that uses adversarial networks in order to estimate a generative model, proposed in Goodfellow et al. (2014). Its main idea is to train two different networks, a discriminator D and a generator G, in a sort of minimax two player game. While generator G generates instances that mimic items from the database, discriminator D classifies instances into real or fake, i.e., if they are items from the database or created by the generator, respectively. The two networks are trained in an adversarial approach, where G is optimized to maximize the probability of D making an error and D minimizes such probability. This training can be made in alternating iterations, by performing k steps of optimization of D and then l (usually one) steps optimizing G. A diagram of a GAN architecture can be seen in Figure 2.7, in which z represents a random vector, L_{GAN} the loss function for G and L_D the loss function for D.

Generator G receives a random input vector of size z, and is trained to produce a transformed instance. Thus, to generate different fake items, one can input different z sized vectors into G. However, in the original work of Goodfellow et al., there is nothing to guide the generator aside from the randomized vector, so we cannot generate the image as we please. Hence, Conditional Generative Adversarial Networks (cGAN) where created to overcome this issue (Mirza and Osindero, 2014). In cGANs, both the generator G and the discriminator D receive an additional input, which can be used to lead the generator into creating a desired image. This way, instead of estimating a generative model, cGAN can be used to learn an image-to-image transformation with pairs of input-output images. The generator G receives the input image and tries to learn how to transform it into the output image, while the discriminator D receives an input-output pair of images and tries to discover if the output image was generated by G or if it comes from the database (i.e.,

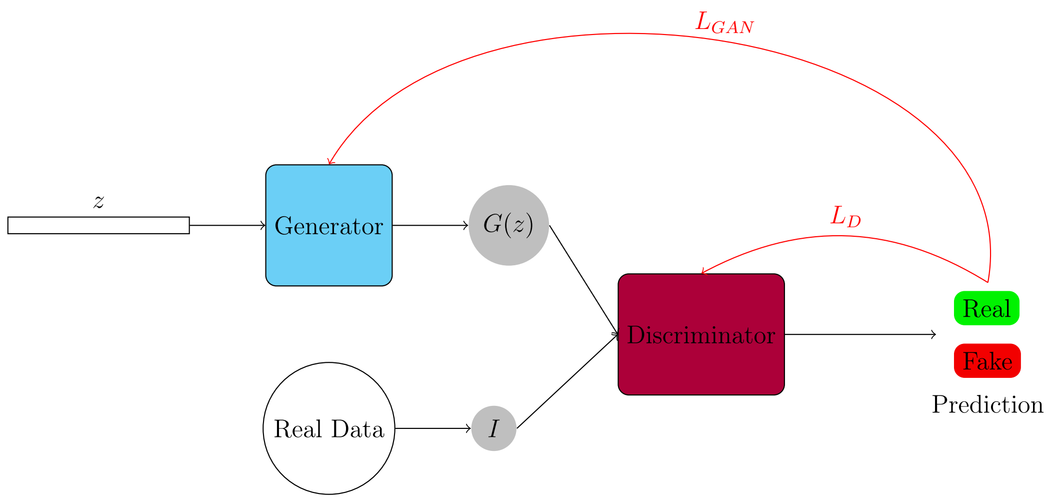


Figure 2.7: *Example of a GAN architecture*

D is a classifier that distinguishes between a real transformation and a learned one).

One of the pioneering methods that uses cGAN to learn an image-to-image transformation was proposed in Isola et al. (2017). Its main idea is that G will learn a mapping from a random vector z and the input image I into the desired output $\Psi(I)$, i.e $G : \{I, z\} \to \Psi(I)$, while D learns to classify a pair of images (I, I') into real or fake (i.e., whether image I' is from the database or generated by G given image I). For example, if this technique is used to learn document binarization, generator G will receive a document image and will try to output a binarized version of it, while D will receive a document image and a binarization, and will try to decide if the binarization was manually made (real) or if it was made by the generator (fake). Therefore, the objective function L_{cGAN} consists of two terms, objective of D and objective of G, and can be expressed as:

$$L_{cGAN}(G, D) = \mathbb{E}_{I,\Psi(I)}[\log(D(I, \Psi(I)))] + \mathbb{E}_{I,z}[\log(1 - D(I, G(I, z)))] \tag{2.5}$$

Thus, L_{cGAN} penalizes the joint configuration of the output. Besides that, the work from Isola et al. also includes a traditional L_1 distance in its final loss, due to previous works finding it beneficial to GAN's performance (Pathak et al., 2016). Therefore, the objective function is a combination of L_1, which performs well enough to low frequency structures, and L_{cGAN}, which models high-frequency structures within a $N \times N$ image patch. As shown by Li and Wand (2016), this discriminator effectively models a Markov Random Field that assumes independence of pixels that are more distant than N pixels.

2.4 SConvNet - A new fully convolutional model

In Sections 2.1 and 2.2 we described the pixel-by-pixel and the patch-to-patch approaches for image-to-image transformations.

Patch-to-patch approaches are fully convolutional. Since convolutional networks commonly use pooling layers, which reduces the input image size, upsampling layers are needed to return the feature map to the original input image size. The problem with this method is that computation of pooling and upsampling feature maps results in pixel location information loss. Therefore, Fully Convolutional Networks for dense prediction use techniques to overcome this problem, such as adding previous encoding layers to upsampled feature maps (FCN, Long et al. (2015)), keeping the indexes of the maximum value in the feature map before the pooling layers (SegNet, Badrinarayanan et al. (2017)) or concatenating previous feature maps to the decoding part of the network (UNet, Ronneberger et al. (2015)). On the other hand, in pixelwise methods there is no such problem, as each pixel output is calculated individually, directly from the information from its surrounding in the input image.

However, regarding processing time, pixelwise approaches are time demanding with respect to prediction and, for large images, it may be a critical issue. On the other hand, patch-to-patch approaches based on fully convolutional networks can predict a large image in only one forward pass, being, therefore, extremely fast in prediction.

In summary, pixelwise methods keep the location information of every pixel, but their prediction is computationally expensive. Patch-to-patch methods, on the other hand, have the opposite behavior. In this section we discuss a simple method that unites the nice characteristics of both approaches.

2.4.1 Concept

Let K be an odd positive integer. A convolution with a $K \times K$ filter outputs a feature map of reduced size, specifically by $K-1$ rows and $K-1$ columns, in comparison to the input size. For instance, a 5×5 convolution applied on a $N \times M$ input will generate a $(N-4) \times (M-4)$ feature map. To generate a feature map of the same size, convolutions are usually applied on a padded version of the input image (input image in which borders of a specific size and specific constant value are added).

However, if convolutions with this mask are applied successively without padding, the resulting maps will have increasingly reduced sizes until it becomes smaller then the mask itself. For instance, if we have an input of size $N \times N$ (N also odd, for convenience) and a mask of size 3×3, we are able to apply a sequence of $\lfloor N/2 \rfloor$ convolutions with this mask, until the result is a single point.

By adding activation layers after each convolution layer, possibly a softmax layer after the last convolution, this architecture can be seen as a single output classifier, that operates on input image patches. An example with an input patch of size 9×9 can be seen in Figure 2.8.

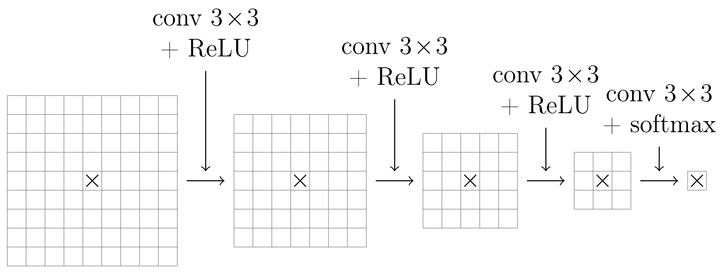


Figure 2.8: *Example of convolution sequence reducing input to a single pixel, acting as a classifier*

Since the above architecture is fully convolutional, it can be applied on input images of arbitrary size. For instance, if we apply the network in Fig. 2.8 on an $N \times M$ input image, the output will be of size $(N - 8) \times (M - 8)$ (since a reduction of 2 rows and 2 columns will occur at each of the four convolution layers). In order to have output images of the same size of the input, it suffices to add a suitable size padding in the input image only. For the example above, a padding of width 4 would be sufficient.

We call this architecture SConvNet (Simple Convolutional Network), as it is a simpler version of other fully convolutional networks. This network can be trained both pixelwise, as in the pixel-by-pixel methods, or patchwise as in the patch-to-patch approaches. The receptive field of the network is the patch in the input image that results at the end in a single pixel. As discussed above, to keep image size, we just need to pad the input image according to the patch size used (for instance, if we use patches of 9×9 and the input image size is originally $N \times M$, padded input image will be of size $(N+8) \times (M+8)$).

The diagram in Figure 2.9 illustrates training of SConvNet following the pixelwise approach, and its applications for dense prediction. Comparing this diagram to the diagram in Figure 2.2, we can understand the difference between pixelwise approaches and SConvNet in its prediction steps. Prediction with pixelwise approaches is done iteratively, predicting the image pixel-by-pixel, while prediction with SConvNet can be done in parallel for every pixel in the image in one forward pass. Note that training can be also done in a patch-to-patch fashion.

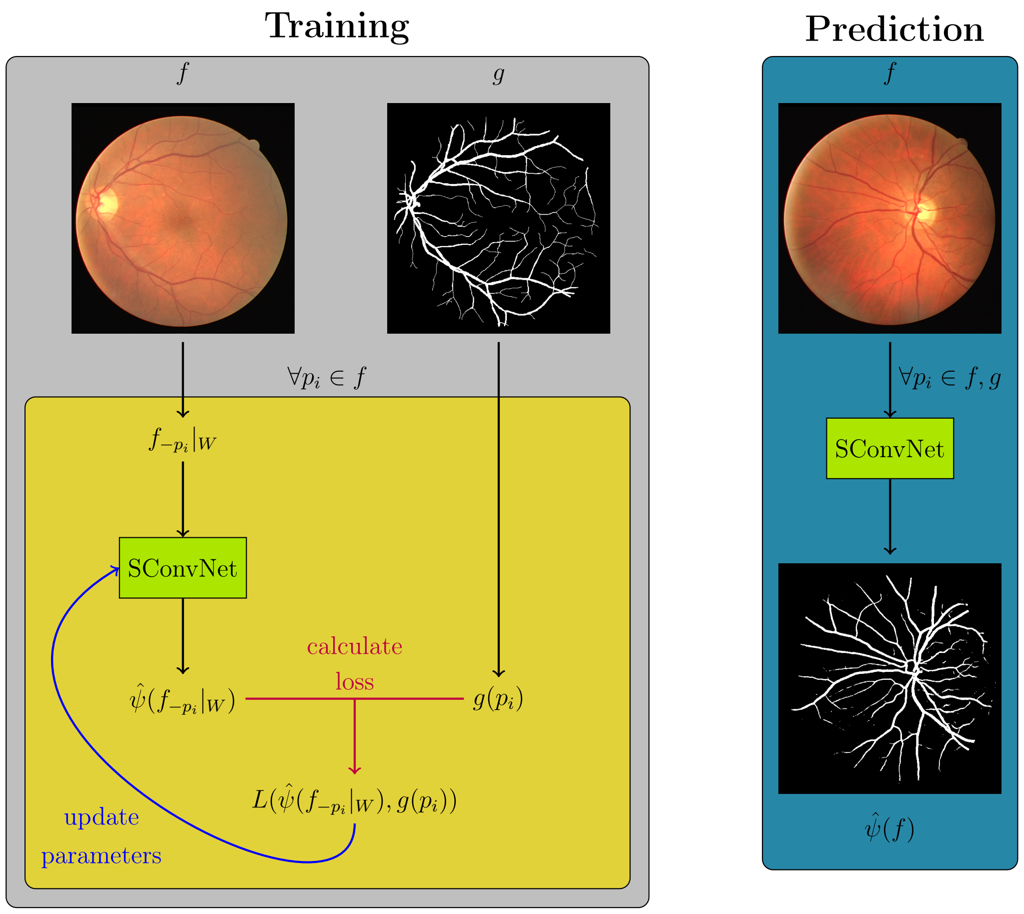


Figure 2.9: *Training and prediction processes of SConvNet. The left side diagram illustrates the training method, in which the yellow block represents a loop over the pixels of the input image, while the right side diagram shows the prediction step.*

2.4.2 Remarks

One of the main differences between SConvNet and other fully convolutional networks such as FCN and UNet is the fact that SConvNet does not employ pooling layers. As a consequence, SConvNet contains more convolutional layers than other architectures with equivalent receptive field. This means that typically the number of weight parameters to be learned may be much larger for SConvNet. Therefore, for large input patch sizes, training SConvNet can become very slow. Nevertheless, there are some strategies that do not rely on pooling layers or convolutions with strides that are useful to reduce the amount of parameters. For instance, dilated convolution (Yu and Koltun (2016)) which is based on sparse kernel matrices, can be used in order to reduce the number of convolutional layers, without reducing the receptive field. Alternatively, the number of feature maps can be reduced in intermediary layers using 1×1 convolutions.

Chapter 3

Evaluation Metrics

Performance of image-to-image transformation methods is usually evaluated by computing a similarity metric between the predicted output and the target image. In this chapter, we describe some metrics that are commonly used for binary images. In Section 3.1 we describe pixelwise evaluation metrics and in Section 3.2, we describe the Skeletal Similarity metric (Yan et al., 2018a), originally proposed to evaluate retinal blood vessel segmentation algorithms, that we also use to evaluate handwritten document binarization algorithms.

3.1 Pixelwise Evaluation Metrics

Pixelwise metrics are calculated by comparing two images pixel-by-pixel. These are the most common metrics for evaluating image transformation algorithms, as they are easier to calculate than structural methods. Most of them are based on statistical measures that are also used for evaluating classification algorithms. From a classification perspective, each pixel is considered as an instance to be classified.

For binary images, we assume value 1 as foreground (also known as positive value) and value 0 as background (also known as negative value).

As is common in two-class classification problems, the metrics are defined based on four main values, which are True Positives (TP, amount of pixels that are positive in the prediction and in the ground truth), True Negatives (TN, amount of pixels that are negative in the prediction and in the ground truth), False Positives (FP, amount of pixels that are positive in the prediction but negative in the ground truth), False Negatives (FN, amount of pixels that are negative in the prediction but positive in the ground truth).

Although Precision and Recall are the most common metrics used for segmentation, we decided to use Sensitivity and Specificity to have a direct correlation with the structural metrics proposed in Yan et al. (2018a). Sensitivity (Se), Specificity (Sp) and Accuracy (Acc) are described in the next subsections.

3.1.1 Sensitivity

Sensitivity, also called Recall or True Positive Rate, is the percentage of positive values that are correctly assigned. It is calculated as follows:

$$Se = \frac{TP}{TP + FN} \tag{3.1}$$

In a medical test to identify a disease in patients, for instance, sensitivity is the test's ability to detect ill patients. Thus, in tests with high sensitivity, a negative result is very helpful to detect healthy patients. Conversely, positive results are less useful, as, although a high sensitivity implies that a good percentage of the ill patients are detected, sensitivity's calculation does not account for wrongly classified positive values (FP). For example, a test that returns positive for every instance would have perfect sensitivity value, despite every negative value being wrongly classified.

In blood vessel segmentation, positive values correspond to vessel pixels, and sensitivity calculates the percentage of vessel pixels that were correctly identified, i.e., the ability of the operator to detect a vessel pixel. In document binarization, positive values correspond to stroke pixels, and sensitivity calculates the percentage of stroke pixels that were found.

3.1.2 Specificity

Specificity, also called True Negative Rate, is the percentage of negative values that are correctly assigned. It is calculated as follows:

$$Sp = \frac{TN}{TN + FP} \tag{3.2}$$

In the medical test example, specificity calculates the test's ability to correctly detect healthy patients. Medical tests with high specificity are useful to correctly identify diseases in the patient, as, similar to sensitivity, specificity calculation does not account for wrongly classified negative values (FN). Thus, a positive value in a high specificity test has a great probability to be correct.

In blood vessel segmentation, negative values account for every non-vessel pixel, and specificity calculates the percentage of these pixels that are correctly assigned. In document binarization, negative value accounts for every non-stroke pixel, and specificity calculates the percentage of these pixels that are correctly identified.

3.1.3 Accuracy

Accuracy is the overall percentage of correctly assigned pixels, and is calculated as follows:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.3}$$

This metric is a more general evaluation of the algorithm's performance. It takes every pixel into account, and calculates how many of the image pixels are correct. Thus, in a balanced set of instances, accuracy is a good overall performance evaluation. However, for unbalanced classes, this metric can be very misleading. For instance, consider a disease that only affects one percent of the population. A medical test that diagnoses every person as healthy (every instance as negative) would have a 99% accuracy, even though the test does not detect any ill patient.

Our two case studies, blood vessel segmentation and document binarization, have very unbalanced classes, as there are much more background pixels than foreground pixels in both.

3.2 Structural Evaluation Metrics

Pixelwise evaluation metrics are very general, and can be used to evaluate algorithms for different image-to-image transformations, such as binarization, segmentation, morphological operations, and so forth. However, as explained in Section 3.1, those metrics can be very misleading, specially in cases with very unbalanced classes. For problems like blood vessel segmentation or document binarization, evaluating the structural consistency between the algorithm prediction and the desired output can be very valuable, giving insights into the algorithm's behaviour (which algorithm better finds the overall vessel structure or which method generates a more readable binary document image).

3.2.1 Skeletal Similarity

Along these lines and to better evaluate algorithms that perform retinal blood vessel segmentation, Skeletal Similarity was proposed by Yan et al. (2018a). This metric compares the structure of the desired ground truth and the given prediction instead of comparing them pixel-by-pixel, hence, better correlates to the visual appearance of the segmentation than pixelwise metrics.

The Skeletal Similarity algorithm is based on comparison of skeletons. First, the skeletons of both images are extracted, and then the skeleton of the ground truth is divided into small segments. For each of those segments, a correspondent set of pixels on the predicted image is found and then a similarity between these pixels and the segment is calculated. The similarity measure in Yan et al. (2018a) combines curve and thickness similarities. A diagram of the algorithm is shown in Figure 3.1 and a more detailed description follows.

As shown in the diagram of Fig. 3.1, I denotes the predicted binary image and I_G the reference ground truth. Skeletal Similarity is calculated according to the following five steps:

1. **Skeletonization of images**

   The first necessary step is to calculate the skeleton of both images I and I_G, which are denoted $Skel$ and $Skel_G$, respectively. As retinal blood vessels are quite simple structures for skeletonization, the thinning method proposed in Suzuki and Keiichi (1987) is used, which works by iteratively removing deletable contour pixels until there is no more deletable pixels.

2. **Decomposition of $Skel_G$ into a set of segments**

   Directly calculating a similarity between two skeletons is a difficult task. Therefore, the skeleton is divided into segments and then similarity is computed segment-wise. To decompose a skeleton, it is separated into multiple segments by cutting it at the branching points. Since the resulting segments usually have distinct length , they are further cut into shorter segments to obtain a more uniform length distribution. This is done by selecting an edge point of a segment and for every 15 connected pixels, making a cut. This divides a segment into smaller segments of length up to 15 pixels. Also, due to the resolution limitation of the images, skeletonization may create false skeleton segments (an example can be seen in Figure 3.2). As most of these false skeleton segments are quite small, a simple heuristic that consists in removing segments that are smaller than a predefined length (for instance, remove

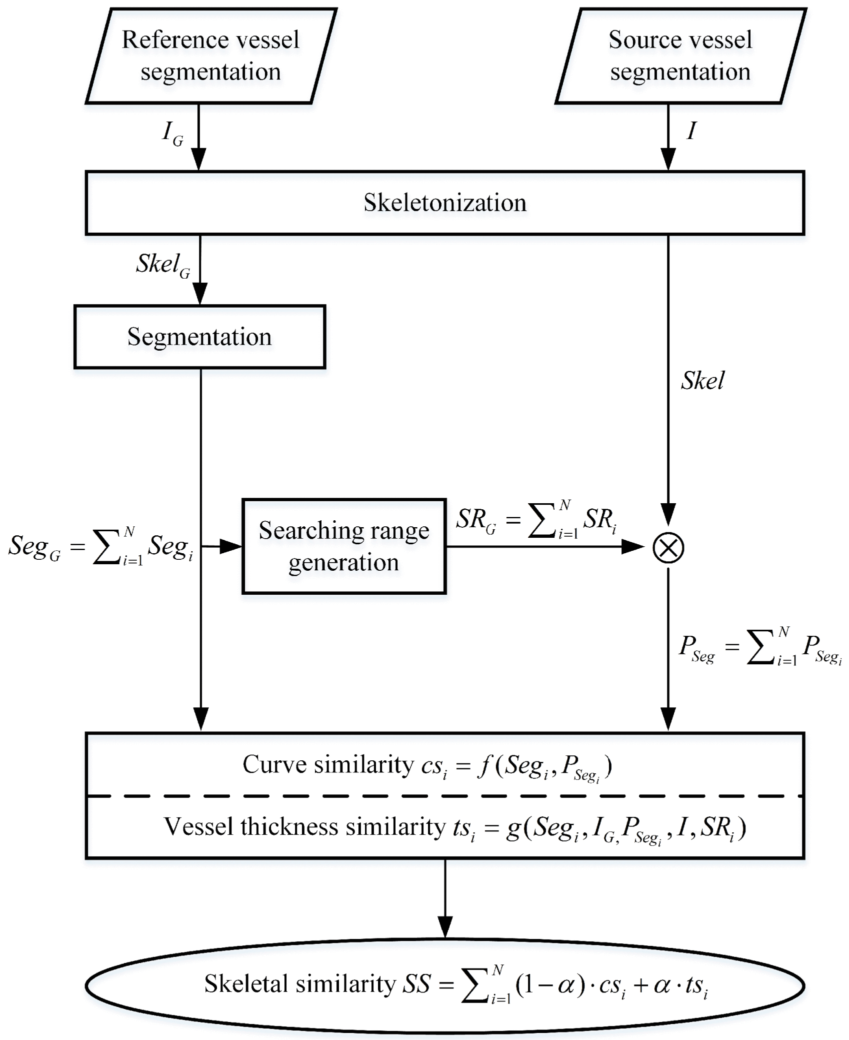

Reference vessel segmentation
Source vessel segmentation
I_G
I
Skeletonization
$Skel_G$
Segmentation
$Skel$
$Seg_G = \sum_{i=1}^{N} Seg_i$
Searching range generation
$SR_G = \sum_{i=1}^{N} SR_i$
$\otimes$
$P_{Seg} = \sum_{i=1}^{N} P_{Seg_i}$
Curve similarity $cs_i = f(Seg_i, P_{Seg_i})$
Vessel thickness similarity $ts_i = g(Seg_i, I_G, P_{Seg_i}, I, SR_i)$
Skeletal similarity $SS = \sum_{i=1}^{N} (1-\alpha)\cdot cs_i + \alpha \cdot ts_i$

Figure 3.1: *Diagram of Skeletal Similarity algorithm (source: Yan et al. (2018a)).*

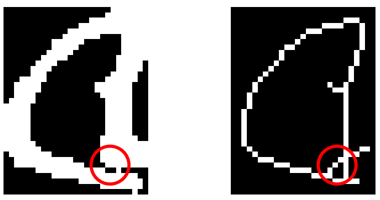


Figure 3.2: *False skeleton segment created due to the limited resolution of the image*

any segment that has length equal to 4 pixels or less) is applied. This segmentation method applied on $Skel_G$ generates a set Seg_G of skeleton segments:

$$Seg_G = \bigcup_{i=1}^{N} Seg_i$$

where N is the total number of segments and Seg_i is the i-*th* skeleton segment.

3. **Matching pixels in $Skel$ to the segments in Seg_G**

   To compare the segments in Seg_G with the skeleton $Skel$, for each $Seg_i \in Seg_G$ a set of the best matching pixels in $Skel$ is calculated. For that, a region denoted searching range SR_i is generated for each segment Seg_i, and a set P_{Seg_i} is created with every pixel in $Skel$ within this region.

   The searching range is generated as follows: for each pixel p in Seg_i, the minimum inscribed circle in I_G centered at p that is fully contained within a vessel is calculated. The diameter of this circle is considered as the thickness t of the vessel at p and the searching radius is defined as:

$$SR_p = \begin{cases} R & \text{if } T_{max} = T_{min} \\ \left\lceil \dfrac{T_{max} - t + \varepsilon}{\lfloor T_{max} - T_{min} \rfloor} \right\rceil \times R & \text{otherwise} \end{cases} \tag{3.4}$$

   where ε is an arbitrarily small positive real number, R is a predefined maximum range, T_{max} is the maximum vessel thickness of the whole image I_G and T_{min} is the minimum vessel thickness of I_G. Thus, the searching range SR_i of a segment Seg_i is defined as the union of the generated searching range for each pixel of the segment, that is, $SR_i = \bigcup_{p \in Seg_i} SR_p$. Then, the searching range for the whole image is represented by the set:

$$SR_G = \bigcup_{i=1}^{N} SR_i$$

For each SR_i, the set P_{Seg_i} of pixels in $Skel$ within SR_i are considered for further calculations, that is, $P_{Seg_i} = SR_i \cap Skel$. The set of all pixels that will be used to calculate Skeletal Similarity is represented by:

$$P_{Seg} = \bigcup_{i=1}^{N} P_{Seg_i}$$

4. **Calculation of Similarities between Seg_i and P_{Seg_i}**

For each segment i $(i = 1, ..., n)$, the similarities are calculated by comparing the pixels in P_{Seg_i} to Seg_i. The similarity is expressed in terms of the curve similarity cs_i and the thickness similarity ts_i.

(a) Curve similarity (cs_i)

Curve similarity cs_i measures the structural consistency between the segment Seg_i and the pixels of P_{Seg_i}, without any thickness measurement.

To calculate the structure similarity of both segments, a curve is fitted to segment Seg_i. Then, the same curve fitting method is applied to the pixels in the set P_{Seg_i}. As the segments are limited to 15 pixels, a cubic function is sufficient. The similarity of both segments is calculated as the similarity between the two fitted curves.

Consider $P_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1$ as the fitted curve of segment Seg_i and $P_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2$ as the fitted curve on the pixels of P_{Seg_i}, and the coefficient vectors $F_1 = (a_1, b_1, c_1)$ and $F_2 = (a_2, b_2, c_2)$. Coefficients d_1 and d_2 are disregarded as they only alter the position of the curve, and do not change its overall structure. Curve similarity between segment Seg_i and P_{Seg_i} is then calculated as:

$$cs_i = \frac{< F_1, F_2 >}{|F_1|.|F_2|} \tag{3.5}$$

where $< ., . >$ is the dot product and $|.|$ measures the length of the vector.

(b) Thickness similarity (ts_i)

Thickness similarity ts_i compares the thickness of segment Seg_i and P_{Seg_i}, measuring how close are the corresponding vessel thickness in images I_G and I.

To compare the thickness of both segments, the same method of measuring thickness to generate the searching range is used (i. e, calculating the diameter of the minimum inscribed circle, in I_G for pixels in Seg_i and in I for pixels in

P_{Seg_i}). The thickness W_{Seg_i} and $W_{P_{Seg_i}}$ are calculated as the average thickness over the set of pixels in Seg_i and P_{Seg_i}, respectively.

As thickness variation in thin vessels is more common than in thick vessels, even in manual annotations, thickness variation in thin vessels are less penalized than in thick ones. Thus, the thickness similarity is defined as:

$$ts_i = \begin{cases} 0 & \text{if } \frac{|W_{Seg_i} - W_{P_{Seg_i}}|}{W_{SR_i}} \geq 1 \\ 1 - \frac{|W_{Seg_i} - W_{P_{Seg_i}}|}{W_{SR_i}} & \text{otherwise} \end{cases} \tag{3.6}$$

where W_{SR_i} is the average thickness of the searching range SR_i.

5. **Skeletal similarity value calculation**

Given the values of cs_i and ts_i, the skeletal similarity value for each segment i is calculated as a weighted sum as:

$$ss_i = (1 - \alpha).cs_i + \alpha.ts_i \tag{3.7}$$

where $\alpha \in [0, 1]$. Thus, the metric can be more easily adapted to different applications.

Also, for segments in I that are very small compared to the corresponding one in I_G, ss_i is set as 0. Very small is defined as percent of Seg_i's length (For instance, $ss_i = 0$ if $|P_{Seg_i}| < 0.6|Seg_i|$). Then, the overall skeleton similarity is calculated by a weighted average of the similarity of the segments, in which the weights are the length of the segment. Thus:

$$SS = \frac{\sum_{Seg_i \in Seg_G} ss_i \times |Seg_i|}{\sum_{Seg_i \in Seg_G} |Seg_i|} \tag{3.8}$$

As the Skeletal Similarity metric is not based on pixel-by-pixel matching, it is necessary to redefine the notion of True Positive, False Positive, True Negative and False Negative. By redefining these, it is possible to calculate sensitivity, specificity and accuracy (the ability of the method to recognize vessels, the percentage of non-vessel segments found, and the overall 'correctness' of the algorithm).

A modified ground truth I'_G is created by assigning the pixels in I_G as P_v and P_{nv}, that is, vessel pixels and non-vessel pixels. As all the pixels in the searching range SR_G are used on the curve similarity calculation, such pixels are denoted as vessel pixels. Furthermore, as every vessel pixel in I_G is used to calculate the thickness similarity, those pixels are also denoted as vessel pixels in I'_G. Thus, vessel pixels P_v are represented by

$(I_G \cup SR_G)$ and every other pixel is counted as non-vessel pixel P_{nv}. Thus, TP, TN, FP, FN can be redefined as:

$$\begin{aligned} TP &= SS \times P_v \\ FN &= (1 - SS) \times P_v \\ FP &= P_{nv}(1) \\ TN &= P_{nv}(0) \end{aligned} \tag{3.9}$$

where $P_{nv}(1)$ represents the number of non-vessel pixels wrongly classified as vessel pixel, and $P_{nv}(0)$ is the number of correctly classified non-vessel pixels in P_{nv}.

Thus, sensitivity, specificity and accuracy can be calculated as:

$$\begin{aligned} rSe &= \frac{TP}{TP + FN} = SS \\ rSp &= \frac{TN}{TN + FP} = \frac{P_{nv}(0)}{P_{nv}} \\ rAcc &= \frac{TP + TN}{P_v + P_{nv}} = \frac{SS \times P_v + P_{nv}(0)}{P_v + P_{nv}} \end{aligned} \tag{3.10}$$

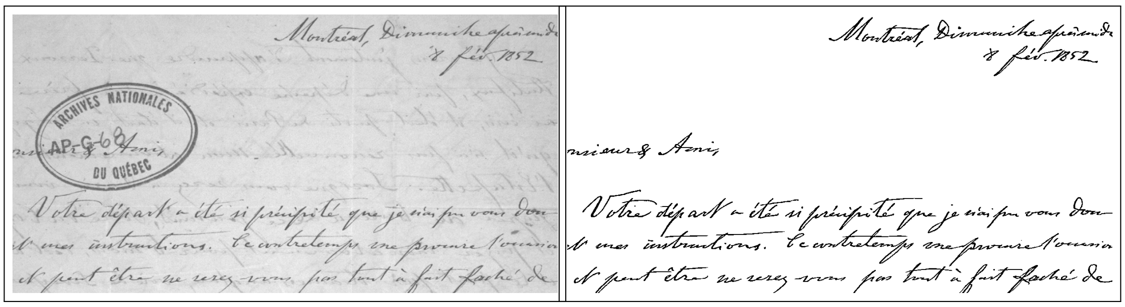
Furthermore, accuracy can be rewritten as a weighted sum of the redefined Sensitivity and the redefined Specificity, as follows:

$$rAcc = \frac{P_v}{P_v + P_{nv}} rSe + \frac{P_{nv}}{P_v + P_{nv}} rSp \tag{3.11}$$

Chapter 4

Skeletal Similarity applied to Document Binarization

As described in the previous chapter, metrics that compare two images pixel-by-pixel can be misleading for high imbalanced data. Binarized document images fall into this category, as there are much more background pixels than foreground ones. This can be seen in Figure 4.1 which is a typical binarized document image.


ARCHIVES NATIONALES
AP-G-68
DU QUÉBEC

(a) *Handwritten document image* **(b)** *Binarized document image*

Figure 4.1: *Example of binarized document image from DIBCO database (Pratikakis et al. (2018))*

Thus, there are many researches on how to better evaluate algorithms for this task, such as the works of Ntirogiannis et al. (2013) and Haiping Lu et al. (2004). However, many of them, such as the ones used on the *Document Image Binarization Competition* (DIBCO, Pratikakis et al. (2018)), are still based on pixel-by-pixel comparison, usually assigning weights to the pixels in order to surpass the problems of the common statistical measures, like sensitivity and specificity.

As the structure of strokes in binarized documents has some similarities with the structure of blood vessels in retinal images, we decided to evaluate Skeletal Similarity, described in Section 3.2.1, to measure stroke structure consistency between two document images. As document binarization algorithms are mainly used to facilitate character recognition, our interest is to evaluate if this metric is able to measure the readability of the binarized image, penalizing structural changes that alters the characters while reducing the impact of small distortions. To that end, we manually generated distorted binary images from a 'perfect' binarization. We used small distortions that do not affect the overall readability of the text, in order to investigate if the Skeletal Similarity is stable across those different distortions. We also generated binarizations with real segmentation algorithms, such as Otsu's binarization or the method from Calvo-Zaragoza and Gallego (2019). Then, we calculated pixelwise and Skeletal Similarity metrics, analyzing the whole images as well as small patches to compare the performance of each metric with the visual quality of the image. For a clearer discussion, we use the following notation:

- pSe - pixelwise sensitivity (Eq. 3.1)
- pSp - pixelwise specificity (Eq. 3.2)
- $pAcc$ - pixelwise accuracy (Eq. 3.3)
- sSe - structural (skeletal similarity) sensitivity (Eq. 3.10)
- sSp - structural (skeletal similarity) specificity
- $sAcc$ - structural (skeletal similarity) accuracy

For character recognition, the thickness of the strokes does not convey much information (there are even character recognition systems that completely discards the thickness of the text). Therefore, we only calculated the curve similarity (setting $\alpha = 0$) and discarded the thickness part of the Skeletal Similarity. We also needed to define a maximum range R to compute Skeletal Similarity, which is dependent on the dataset used. In our experiments, we tried different R values, however, due to the heterogeneous nature of DIBCO dataset, larger values allowed unwanted structural changes while smaller values were not stable to small distortions. We found that setting $R = 2$ offered the best balance between penalizing structural changes and forgiving small pixel alterations.

Section 4.1 presents the experiments made with manually generated distorted images to check if the metrics are consistent with small distortions, while Section 4.2 reports our experiments with real images generated by document binarization algorithms.

4.1 Evaluation on simulated data

To create the manually distorted images, we used 31 binarized images from the Synchromedia Multispectral Dataset (S-MS) database (Hedjam and Cheriet (2013a,b)). S-MS is a public database that was part of a contest from *International Conference on Document Analysis and Recognition 2015 (ICDAR 2015).* It consists, as of the time these experiments were made, of 31 multispectral document images collected from the Bibliothèque et Archives nationales du Québec (BAnQ).

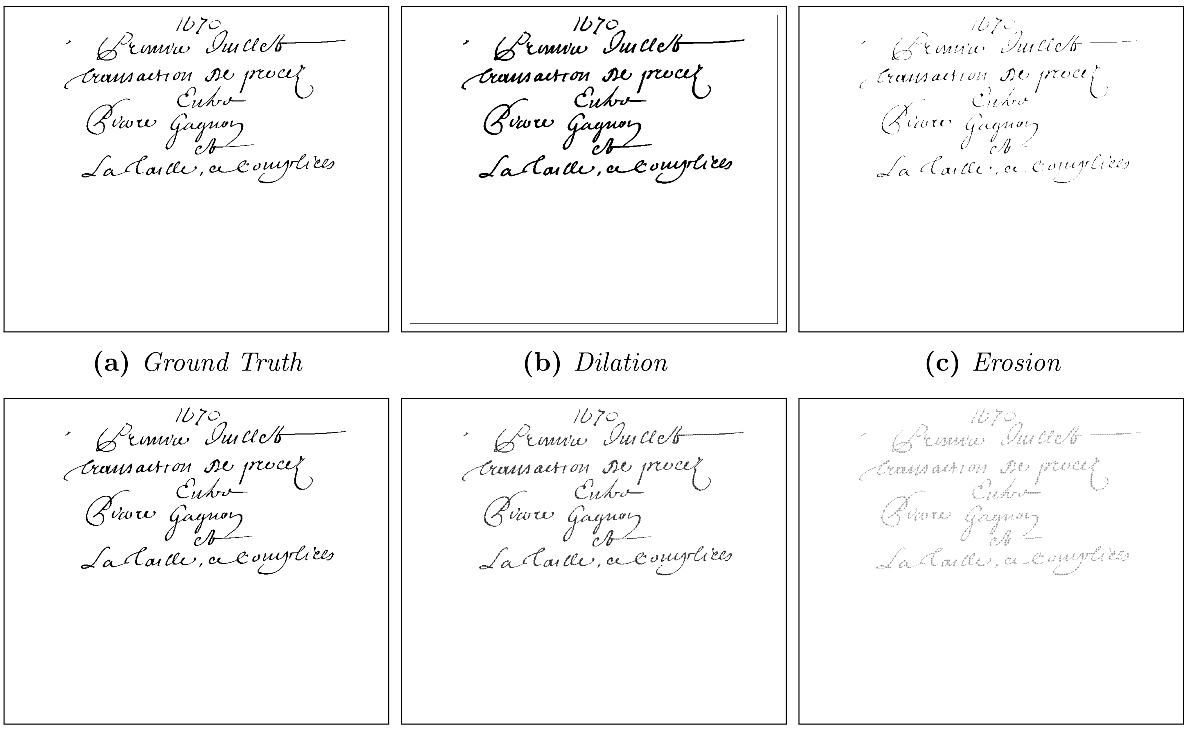
We create five different distortions for each image by applying the following transformations:

- **Morphological Dilation and Erosion:** chosen to test the Skeletal Similarity metrics stability for thickness differences.
- **2-pixel Translation:** Each pixel in the distorted image has the value of the pixel two columns behind and two rows above in the original image. This is a nearly imperceptible transformation that greatly affects pixelwise metrics.
- **Missing points:** Every pixel on an even column and even row was set to background. This was generated to analyse how the metrics are affected by noise in the image.
- **Missing lines:** Even columns and even rows were set to background. This was created to investigate the metrics stability for information loss.

An example for each transformation is shown in Figure 4.2. As we see in Figure 4.2, the chosen distortions do not significantly alter the overall structure of the text, even though in some cases, such as missing rows and columns, the text becomes less visible. Some distortions, such as dilation or erosion can alter the structure of some characters (such as thin strokes vanishing in erosion or separate near strokes becoming one in dilation), however the alteration is small enough and the majority of the text remains readable.

It is important to note that, although the text in Figure 4.2f appears to be invisible at a first glance, this is because the image is shown in reduced resolution. Figure 4.3 displays an image patch from Figure 4.2f with a larger resolution, illustrating that albeit the distortion, the readability is kept.

Table 4.1 compares pixelwise against Skeletal Similarity metrics for the images in Figure 4.2. As expected, sSe is way more stable across the distortions, with its value



(a) *Ground Truth* **(b)** *Dilation* **(c)** *Erosion*

(d) *Displacement* **(e)** *Noise* **(f)** *Missing rows and columns*

Figure 4.2: *Examples of manually generated distorted images. See details in the text*

Table 4.1: *Pixelwise and SS metrics for the images in Fig. 4.2*

	Sensitivity		Specificity		Accuracy	
Distortion	*pSe*	*sSe*	*pSp*	*sSp*	*pAcc*	*sAcc*
Dilation	**100**	98.3	98.1	**99.3**	98.2	**99.3**
Erosion	45.4	**70.3**	**100**	**100**	**98.6**	**98.6**
Displacement	36.3	**91.8**	98.4	**99.2**	96.8	**98.9**
Missing Points	75	**98.8**	**100**	**100**	99.4	**99.9**
Missing Lines	25.1	**87.1**	**100**	**100**	98.1	**99.4**

ranging from 70.3 to 98.8, while pSe values range from 25.1 to 100. As sensitivity measures the amount of correctly assigned positive instances, pSe value of 36.3 on the displaced image (Fig. 4.2d) is the same of an image with about two thirds of the strokes missing, even though the displaced image is perceptually very similar to the original one. The same happens for the Missing Rows and Columns distortion (Fig. 4.2f), where pSe value is the same as if only one quarter of the text was correctly assigned, even though the main structure of the whole text still remains.

We also calculated the average pixelwise and structural metrics for the entire database, and the values are shown in Table 4.2. Note that the same observed pattern in Table 4.1 occurs with respect to the average metrics, where sSe is more stable than pSe. Also, displaced images and missing rows and columns images have a pSe value that is equal as

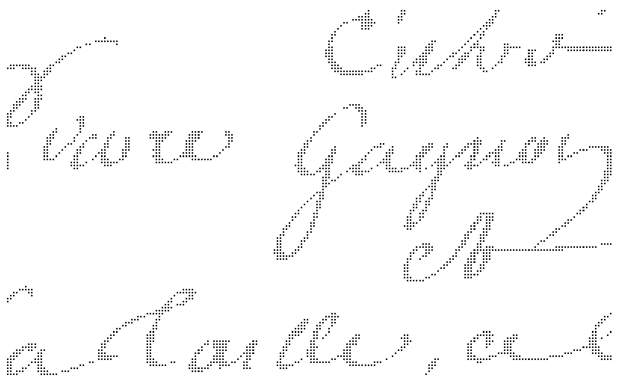


Figure 4.3: *Zoom in of a patch from the image in Fig. 4.2f, illustrating that the distortion does not affect the overall structure*

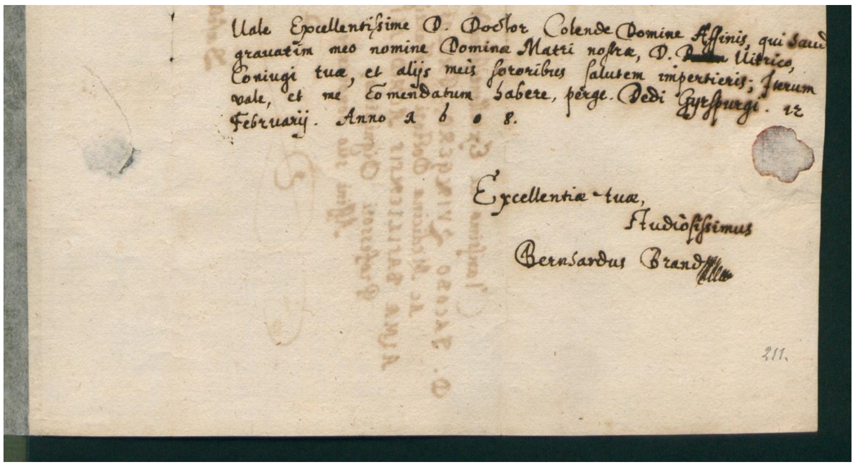
when only half or a quarter of the text is recovered, respectively. Thus, sSe better represents the readability of the binary image as well as the structural consistency between the prediction and the target.

Table 4.2: *Average metrics computed on the 31 manually distorted images.*

	Sensitivity		Specificity		Accuracy	
Distortion	pSe	sSe	pSp	sSp	$pAcc$	$sAcc$
Dilation	**100**	97.5	95.8	**98.6**	96.2	**98.5**
Erosion	49.2	**78.1**	**100**	**100**	96.8	**97.6**
Displacement	53.3	**91.5**	93.7	**95.3**	91.3	**94.9**
Missing Points	74.9	**98.9**	**97.5**	**97.5**	96.1	**97.8**
Missing Rows	24.9	**90.3**	**99.1**	**99.1**	94.3	**98.3**

4.2 Evaluation on Real Data

To evaluate how Skeletal Similarity behaves as a document image binarization evaluation metric, we selected four binarization algorithms and applied them on the images from *Document Image Binarization Competition 2018 (DIBCO)* (Pratikakis et al., 2018) and then compared the binarized images with respective ground-truth binarizations. The algorithms, namely Otsu (Otsu, 1979), Sauvola (Sauvola and Pietikäinen, 2000), Howe (Howe, 2011) and Zaragosa (Calvo-Zaragoza and Gallego, 2019), were chosen based on their expected performance differences. For instance, Otsu is a well-known simple binarization algorithm, while Sauvola is similar to Otsu but was designed specifically for document image binarization. Howe was the binarization part of the best performing method in DIBCO 2018, while Zaragosa is a recent deep learning state-of-the-art document binarization algorithm. Thus, we could investigate the Skeletal Similarity in the same image



Vale Excellentissime D. Doctor Colende Domine Affinis, qui haud
grauatim meo nomine Dominæ Matri nostræ, D. [illegible] Vitrico,
Coniugi tuæ, et alijs meis sororibus salutem impertieris; iterum
vale, et me comendatum habere, perge. Dedi Gyrspurgi. 12
Februarij. Anno 1 6 0 8.

Excellentiæ tuæ,
Studiosissimus
Bernhardus Brand[illegible]

211.

Figure 4.4: *Image from the DIBCO 2018 released set*

with different binarization performances. A brief description of each algorithm is presented at the Appendix A.

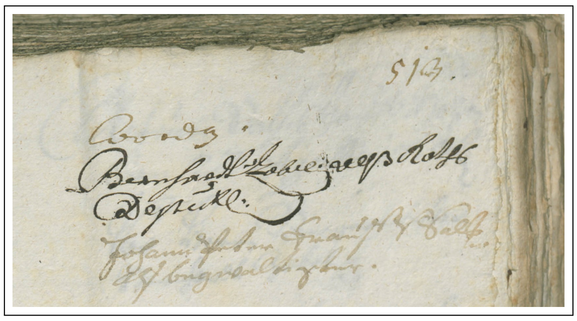
DIBCO is an annually organized competition that benchmarks the state-of-the-art algorithms in document binarization. The organizers release a set of new images each year, which consists of scanned document images that contains representative degradations that are challenging for binarization algorithms, and, later in the same year, a manually generated ground truth of those images alongside the results of the submitted methods are released. Thus, participants can train and test their methods using the images from previous years, apply their methods on the released images and then submit the results to evaluation. In 2018, the released set of images consisted of 10 handwritten document images of varying sizes. One of these 10 images is shown in Figure 4.4.

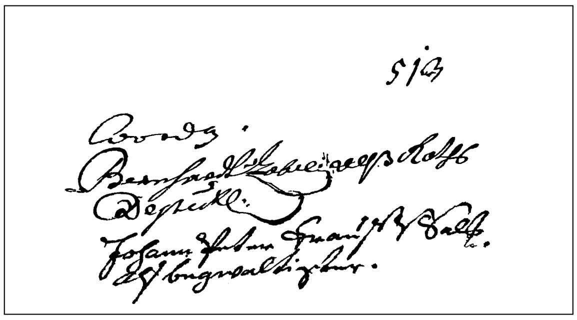
Among the four algorithms we have selected, only Zaragosa's method requires training. We trained Zaragosa's method with DIBCO images from previous years. All methods were applied to images from DIBCO 2018 dataset.

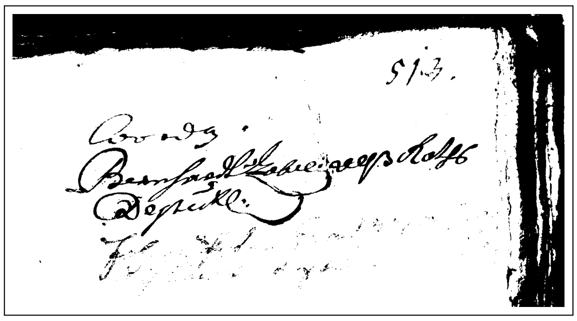
4.2.1 Results and Discussion

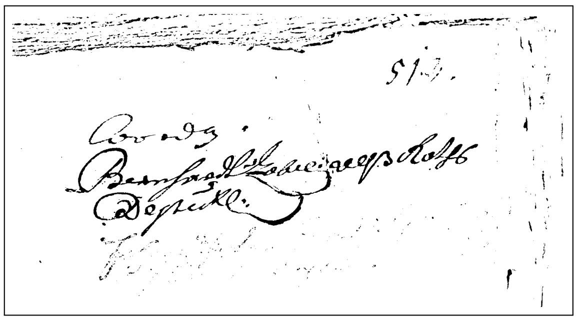
In this section we present results and discussions regarding the binarizations by the four algorithms. Similarly to the case of simulated images (Fig. 4.1), we compute the pixelwise and structural metrics between the binarized images and the corresponding ground-truth images.

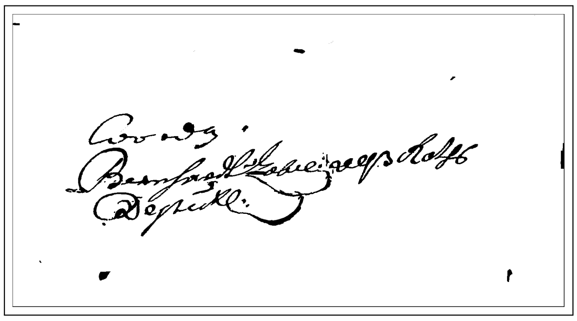
We first compare these metrics with respect to one image in DIBCO 2018. The selected original image, the respective ground-truth binarization, and the results by the four binarization algorithms are shown in Figure 4.5. The metrics for the binarization algorithm results are presented in Table 4.3.


(a) *Input*


(b) *Ground truth binarization*


(c) *Otsu's method*


(d) *Sauvola's method*


(e) *Howe's method*

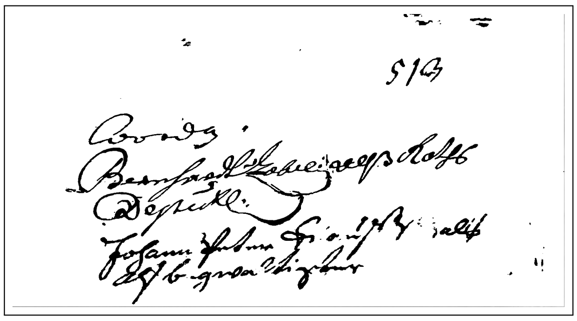

(f) *Zaragosa's method*

Figure 4.5: *Example of an image binarized by the four chosen methods*

Differently from the manually generated images, visual inspection of the real binarizations is not straightforward. As it can be seen in Figure 4.5, while the superiority of Zaragosa's binarization is quite clear (as correctly indicated by the metrics shown in Table 4.3), comparison amongst the other binarizations is rather subjective. For example, take Sauvola's and Howe's results. Howe's binary image is clearer than Sauvola's, having less noise and better defined strokes. Nevertheless, Sauvola's method correctly assigns most pixels of the characters on the top-right corner of the image, which are completely missed by Howe's binarization. Furthermore, strokes with less contrast, such as the last two lines of the document, are completely lost in Howe's image, while in Sauvola's at least some pixels are correctly assigned. This appears to be accounted for in the calculated metrics by the difference between the pixelwise and Skeletal Similarity metrics. On the one

Table 4.3: *Skeletal Similarity metrics calculated on images of Figure 4.5*

	Sensitivity		Specificity		Accuracy	
Algorithm	pSe	sSe	pSp	sSp	$pAcc$	$sAcc$
Otsu	53.8	62.5	81.9	81.8	80.0	80.4
Sauvola	44.1	58.1	98.5	98.5	94.8	95.5
Howe	47.0	51.6	99.3	99.3	95.7	95.7
Zaragosa	87.2	90.2	99.5	99.5	98.6	98.8

hand, Howe's pSe is higher than Sauvola's, but on the other hand Sauvola's sSe is higher than Howe's. In this situation, analyzing both metrics can be beneficial to compare the performance of each method.

In order to further investigate Skeletal Similarity metrics, we selected small image patches with interesting differences between the binarizations, and analyzed both metrics calculated on them. Examples of such patches can be seen in Figures 4.6, 4.7 and 4.8, and the calculated metrics for those examples are shown in Tables 4.4, 4.5 and 4.6. Patches were chosen according to common problems of binarization algorithms, such as information loss, misleading results due to small differences or structural differences that could induce errors in a character recognition algorithm. Other selected patches, all binarized images and the pixelwise and structural metrics calculated in all images and patches are available on the following website: https://imageu.github.io/data+code/. Below we show three examples of the selected patches and discuss the calculated metrics on them.

Figure 4.6 shows a patch in which each algorithm shows a different case of missing information. Otsu's method, although correctly assigning every stroke pixel, assigns almost all of the pixels in the image as stroke pixels, which generates a perfect value on pSe but a poor pSp. Sauvola's method, although with noise and some broken structures, is the best at recovering the strokes structure, which is shown by sSe value being the highest amongst all others. Howe's method completely misses an entire word, however its pSe value is higher than Sauvola's. Zaragosa's binary image, that presents the best pSe value with the exception of Otsu's outlier, has the best well defined strokes, yet, it misses a structure that could be very important in character recognition (the lasso in the beginning of the second word). Values of $pAcc$ and $sAcc$ showcase a similar pattern, with Sauvola's being the best performing algorithm with respect to $sAcc$ due to the most recovered stroke structure, and Zaragosa's the best in $pAcc$ due to its superiority in the pixel level (better defined strokes, without much noise or gaps in the stroke).

Figure 4.7 showcases an example of the stability of Skeletal Similarity metrics for characters with thickness differences, which contrasts with pixelwise metrics values. In all of the images, the strokes are visually well defined and the readability of the text remains

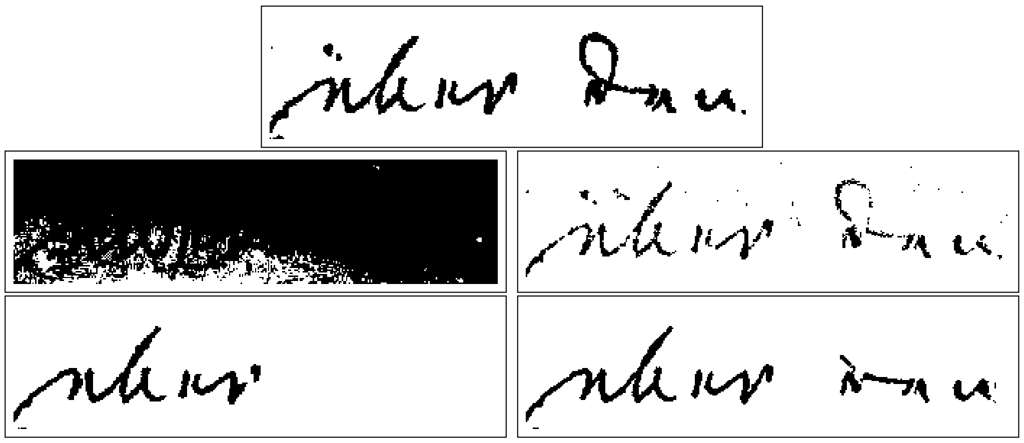


Figure 4.6: *Patch with missing information. From left to right, top to bottom: Otsu's, Sauvola's, Howe's and Zaragosa's methods*

Table 4.4: *Skeletal Similarity and Pixelwise metrics calculated on patches of Fig 4.6*

	Sensitivity		Specificity		Accuracy	
Algorithm	*pSe*	*sSe*	*pSp*	*sSp*	*pAcc*	*sAcc*
Otsu	100	52.9	10.9	11.1	20.0	16.9
Sauvola	55.4	88.6	99.8	99.8	95.3	98.2
Howe	50.2	54.1	99.9	100	94.8	93.6
Zaragosa	69.9	82.9	99.9	100	96.8	97.6

intact. However, due to small thickness differences, *pSe* values range from 77.1 to 89.1 and *pAcc* values range from 95.4 to 97.1. *sSe* is way more stable, with all metric values at perfect value (except Zaragosa's *sSe*, which has an almost irrelevant difference of 0.2 per cent).

Figure 4.8 displays an example of structural change being better captured by Skeletal Similarity metrics than by pixelwise metrics. Looking at the last valley like region, on the right side of the image, Otsu's algorithm completely fills it, changing the overall structure of the character (what was similar to the character 'u', now looks like a filled character 'a', a big dot, a filled circle, and so on). Comparing this image with Sauvola's result, it is clear that the latter is a better binarization and is more similar to the ground truth, however that is not accurately captured in the pixelwise metrics. On the other hand, Skeletal Similarity metrics correctly shows this superiority, specially if we compare sensitivity values (pSe = 90.1 and sSe = 63.7 on Otsu's image, while in Sauvola's we have pSe = 88.8 and sSe = 84.2).

Our experiments have shown that Skeletal Similarity metrics have a better correlation to the overall structure of the binarized image than pixelwise metrics, better evaluating structural quality of the output image. These metrics are more stable to small thickness differences and are more compliant to pixel alterations that does not affect

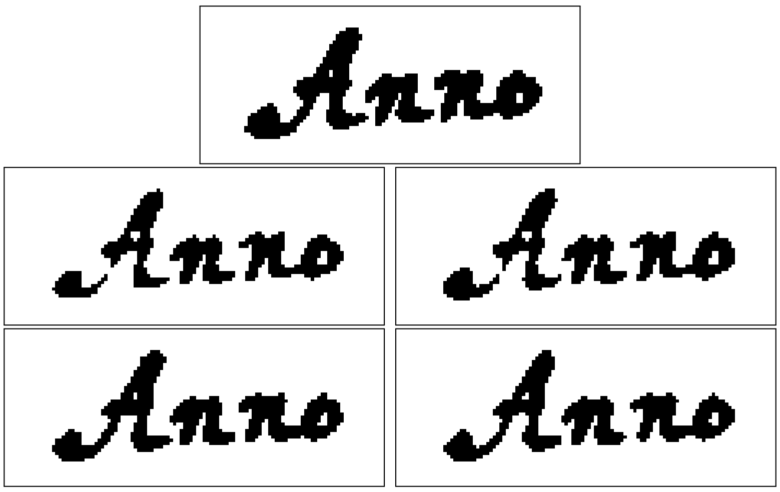


Figure 4.7: *Patch with small thickness differences. From left to right, top to bottom: Otsu's, Sauvola's, Howe's and Zaragosa's methods*

Table 4.5: *Skeletal Similarity and Pixelwise metrics calculated on Fig. 4.7*

	Sensitivity		Specificity		Accuracy	
Algorithm	*pSe*	*sSe*	*pSe*	*sSe*	*pSe*	*sSe*
Otsu	77.1	100	100	100	95.4	100
Sauvola	82.3	100	100	100	96.4	100
Howe	89.1	100	99.9	100	97.7	100
Zaragosa	85.6	99.8	100	100	97.1	100

the visual appearance of the image. Moreover, these metrics heavily penalize structural changes, such as the formation of blobs or filled holes. Therefore, Skeletal Similarity metrics can be used for evaluating the structure consistency between a manual binarization of a document image and a prediction from a binarization algorithm.

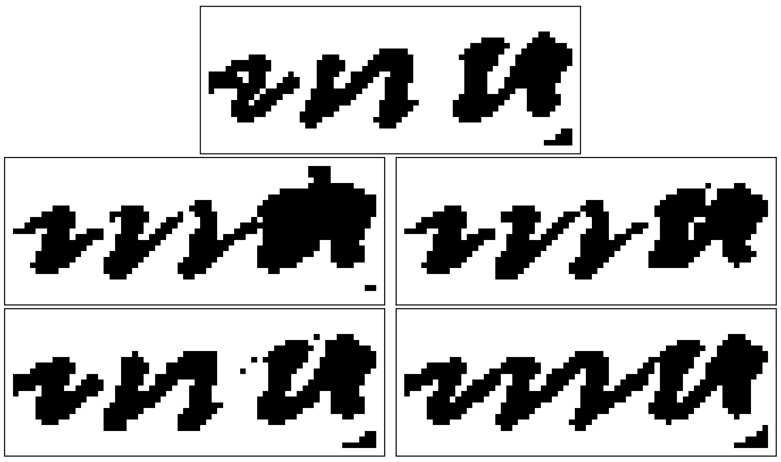


Figure 4.8: *Patch with small structural difference. From left to right, top to bottom: Otsu's, Sauvola's, Howe's and Zaragosa's methods*

Table 4.6: *Skeletal Similarity and Pixelwise metrics calculated on Fig. 4.8*

	Sensitivity		Specificity		Accuracy	
Algorithm	*pSe*	*sSe*	*pSe*	*sSe*	*pSe*	*sSe*
Otsu	90.1	63.7	89.3	93.8	89.5	79.9
Sauvola	88.8	84.2	93.7	97.5	92.2	91.3
Howe	98.5	99.9	96.2	98.7	96.9	99.3
Zaragosa	94.1	99.9	94.1	96.8	94.1	98.2

Chapter 5

Experimental Results

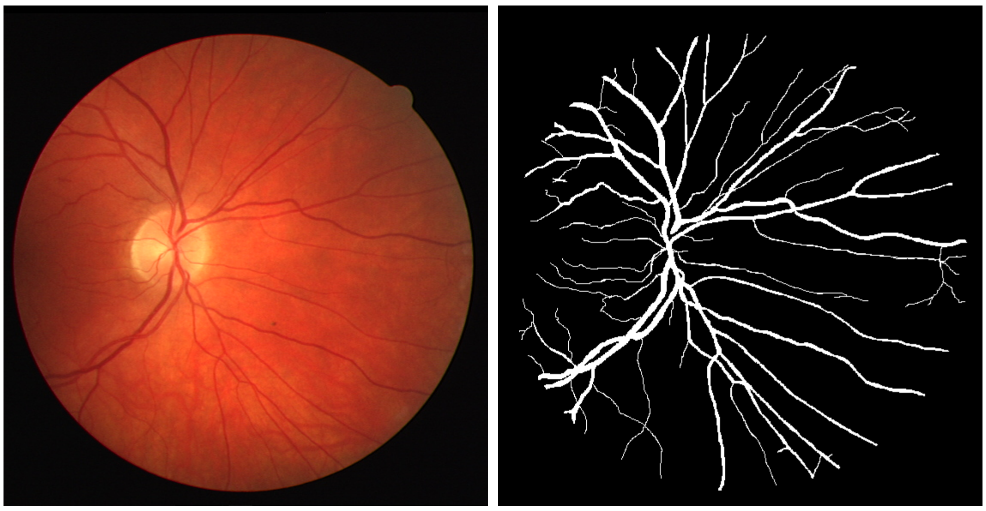
In this chapter we present results regarding the evaluation of some of the methods described in Chapter 2 on two image processing tasks, namely segmentation of blood vessels in retinal images and binarization of handwritten document images. As shown in Chapter 4, skeletal similarity metrics, originally proposed for evaluating blood vessel segmentation, are also adequate to measure the structural consistency of binarized handwritten documents, in a complementary way to traditional pixelwise metrics. Therefore, to compare the methods, we compute both pixelwise and skeletal similarity metrics (the ones that are described in Chapter 3).

5.1 Experiment Setup

In this section we describe the datasets used in the experiments and also implementation related details of the tested methods.

5.1.1 Datasets

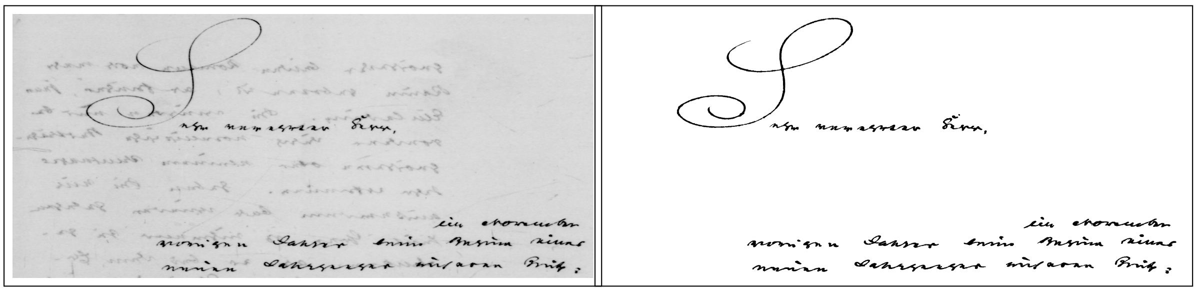
DRIVE (Digital Retinal Images for Vessel Extraction) is a curated dataset introduced in Staal et al. (2004), with the goal of enabling comparative studies of segmentation algorithms. The dataset contains 40 retinal images obtained from a diabetic retinopathy screening program in The Netherlands. They are divided into training and test sets, each containing 20 images. Each image in the training set was manually segmented by experienced ophthalmologists, while every image in the test set was manually segmented by professionals (which can be used as ground truth) and by non-professionals (which can be considered as a benchmark to compare the algorithms). We used only the first set. All original images are colored and of size 565 × 584 pixels. Figure 5.1 shows an example of an input-output pair of images of this dataset.



(a) *Retinal image* **(b)** *Segmentation of blood vessels*

Figure 5.1: *Input-output pair of images from DRIVE dataset.*

DIBCO is a dataset used for evaluating binarization algorithms, it is the same dataset used in the experiments of Chapter 4. Document binarization, and especially of handwritten old documents, is challenging due to degradations such as the presence of stain or watermarks, or color and contrast patterns that resemble strokes. Figure 5.2 shows an image from DIBCO dataset. Note that, although we consider foreground pixels as white (value 1) and background pixels as black (value 0) as it is commonly assumed, we print the binarized images with inverted colors for better visualization.



(a) *Handwritten document image* **(b)** *Binarization of text strokes*

Figure 5.2: *A pair of input-output images from DIBCO dataset.*

These two datasets were chosen based on the following criteria:

- **Local Definition:** This is one of the properties of image-to-image transformations described in Section 2.1 (Eq. 2.2). We chose tasks where the assumption of such property is reasonable. In fact, both for DIBCO and DRIVE, it is possible in general to decide if a pixel is positive or negative by just analyzing pixels in a small region around the pixel, as highlighted in Figures 5.3 and 5.4.

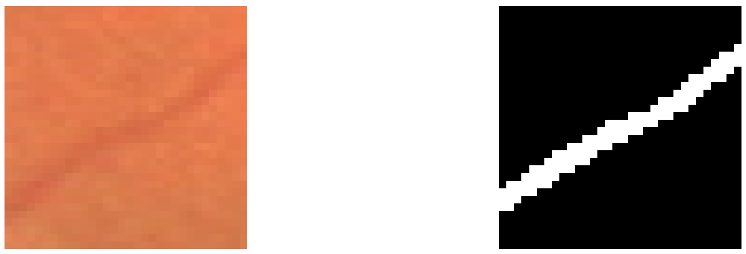


Figure 5.3: *Patch of an input image from DRIVE dataset and the corresponding segmentation. Although the patch restricts content to only a local information, one can segment the vessels fairly well.*

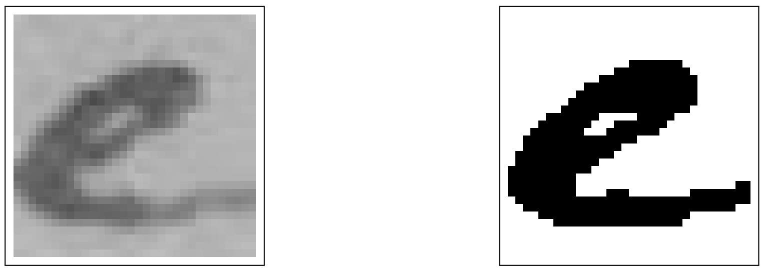


Figure 5.4: *Patch of an input image from DIBCO dataset and the corresponding binarization. Fairly good binarization can be computed even when only a small region is examined.*

- **Similar structure:** As one of the interests in this work is related to structural aspects of the images, and specifically on how well image-to-image transformation methods preserve structural information, we chose the two datasets taking into consideration that in both the objects of interest have a connected, thin and elongated structure. This way, eventual intrinsic differences between methods may be better observed.

- **Public availability:** It facilitates the reproduction of the experiments. Both datasets are publicly available. DRIVE dataset is available in the following link: https://drive.grand-challenge.org/. DIBCO dataset is accessible from the page of the 2018 competition (https://vc.ee.duth.gr/h-dibco2018/). It contains the links for downloading the images of the previous years, as well as an evaluation tool used in the competition.

- **Different aims:** Although images in both datasets share some structural similarity, there are distinct challenges and goals. In retinal blood vessel segmentation, the most challenging part of the task is detecting thin and small vessels. On the other hand, in document binarization, the thin and small strokes may be missed as long as the main structure of the strokes are preserved; the greatest challenges are related with document degradations, such as stains and watermarks.

The datasets were partitioned into training, validation and testing sets as follows:

DIBCO	Training:	87 images (from the competitions held between 2009 to 2016)
	Validation:	20 images (from the competition held in 2017)
	Testing:	10 images (from the competition held in 2018)
DRIVE	Training:	16 images (first 16 images from the original training set)
	Validation:	4 images (last 4 images from the original training set)
	Testing:	20 images (the same as provided by the dataset)

5.1.2 Evaluated Methods

To perform the experiments, we selected instances of the methods described in Chapter 2. For each method we fixed a network architecture, as listed below.

- **CNN:** (Section 2.1.1) The architecture consists of two convolutional layers with 32 and 64 filters, respectively. Each one followed by a pooling layer and ReLU activation. Then, two fully connected layers with 1024 and 2 filters, respectively, followed by a sigmoid activation function. This architecture is trained in a sliding window manner, extracting a 13×13 patch from the input image and the value of the central pixel in the output image.

- **FCN:** (Section 2.2.1) The architecture consists of two convolutional layers with 64 filters each, followed by a max-pooling layer, then two more convolutional layers with 128 filters each, followed by a max-pooling layer, then an upsampling layer and a final softmax layer. Each convolutional layer is followed by a ReLU activation. In our experiments we used the most basic model, without concatenation or fusion of shallow layers.

- **UNet:** (Section 2.2.2) The encoder path consists of two convolutional layers (with 32 filters each), followed by a max-pooling layer, then two more convolutional layers (with 64 filters each) and one max-pooling layer. The decoder path consists of a similar structure, with an upsampling layer followed by two convolutional (64 filters), then another upsampling and two convolutional layers (32 filters). Between the two paths there are two convolutional layers (128 filters) and in the end of the decoder path there is a last convolutional layer followed by a softmax activation function. Each convolutional layer, except the last one, is followed by a ReLU activation.

- **SConvNet:** (Section 2.4) It consists of a sequence of 14 convolutional layers (respectively with $8, 8, 8, 16, 16, 16, 32, 32, 32, 64, 64, 64, 128, 128$ filters) and a final softmax layer. This corresponds to a receptive field of size 29×29. Each convolutional layer is followed by a ReLU activation.

- **U-GAN** and **S-GAN**: (Section 2.3.2) These are GAN architectures where the generative part is either the architecture of UNet described above (U-GAN) or the architecture of SConvNet above (S-GAN). The discriminator in both cases consists of a basic convolutional network with 4 convolutional, batch normalization, ReLU activation layers and a final sigmoid activation function. The input of the discriminator is a concatenation between the input image and output image (real target from the original training data or target generate by the generator). U-GAN was created as an effort to further analyze the results on DIBCO, due to the large deviation of the other methods.

These models were trained using patches from images of the training set and validated on patches extracted from the images in the validation set. Some of the training hyperparameters were fixed whereas others were adjusted to the architecture as well as to the datasets. The fixed parameters, used for the training of all models, are:

- Adam Optimizer
- Cross-Entropy as loss function
- Scaling the inputs to have zero mean and unit variance
- Threshold of 0.5 (Output pixels higher than 0.5 were set to foreground, background otherwise)

Besides these fixed ones, the other hyperparameters that have been adjusted for each model and dataset are summarized in Tables 5.1 and 5.2, for DRIVE and DIBCO, respectively.

Some of the differences between hyperparameters were due to model requirements. For instance, to return a same sized output, FCN and UNet must have an input size that is multiple of 2^k, where k is the number of pooling/upsampling layers. To be able to reduce the patch to a 1×1 output, the input image size of SConvNet must be odd. The difference in the number of epochs difference were due to efficiency and convergence.

Table 5.1: *Hyperparameters for training the model on DRIVE.*

	Patch size		Batch		
Method	Input	Output	size	Epochs	Learning rates
CNN	13×13	1×1	256	10	best of $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$
FCN	32×32	32×32	32	60	10^{-4}
UNet	32×32	32×32	32	60	10^{-4}
SConvNet	29×29	1×1	128	2	10^{-4}
S-GAN	35×35	7×7	256	50	$D : 10^{-4}$ $G : 4.10^{-4}$

Table 5.2: *Hyperparameters for training the model on DIBCO.*

	Patch size		Batch		
Method	Input	Output	size	Epochs	Learning rates
CNN	13×13	1×1	256	10	best of $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$
FCN	32×32	32×32	32	60	10^{-4}
UNet	32×32	32×32	32	60	10^{-4}
SConvNet	29×29	1×1	128	2	10^{-4}
U-GAN	48×48	48×48	32	101	$D : 10^{-4}$ $G : 4.10^{-4}$
S-GAN	61×61	25×25	128	101	$D : 10^{-4}$ $G : 4.10^{-4}$

SConvNet and CNN achieved convergence with a few epochs, but the time to train each epoch was larger (due to the larger amount of parameters in SConvNet and larger training data in CNN). Conversely, the speed of each training epoch for FCN and UNet was faster, but these architectures required more epochs to achieve convergence. As U-GAN was only created to further analyze DIBCO, this architecture was not used on the DRIVE dataset.

Although these methods could be optimized to achieve better performance in each dataset, for instance performing a wider search on hyperparameter space or using training strategies such as data augmentation, we decided to produce a working model for the limited training configuration above, and focus on evaluating the behavior of the methods with respect to structural information preservation.

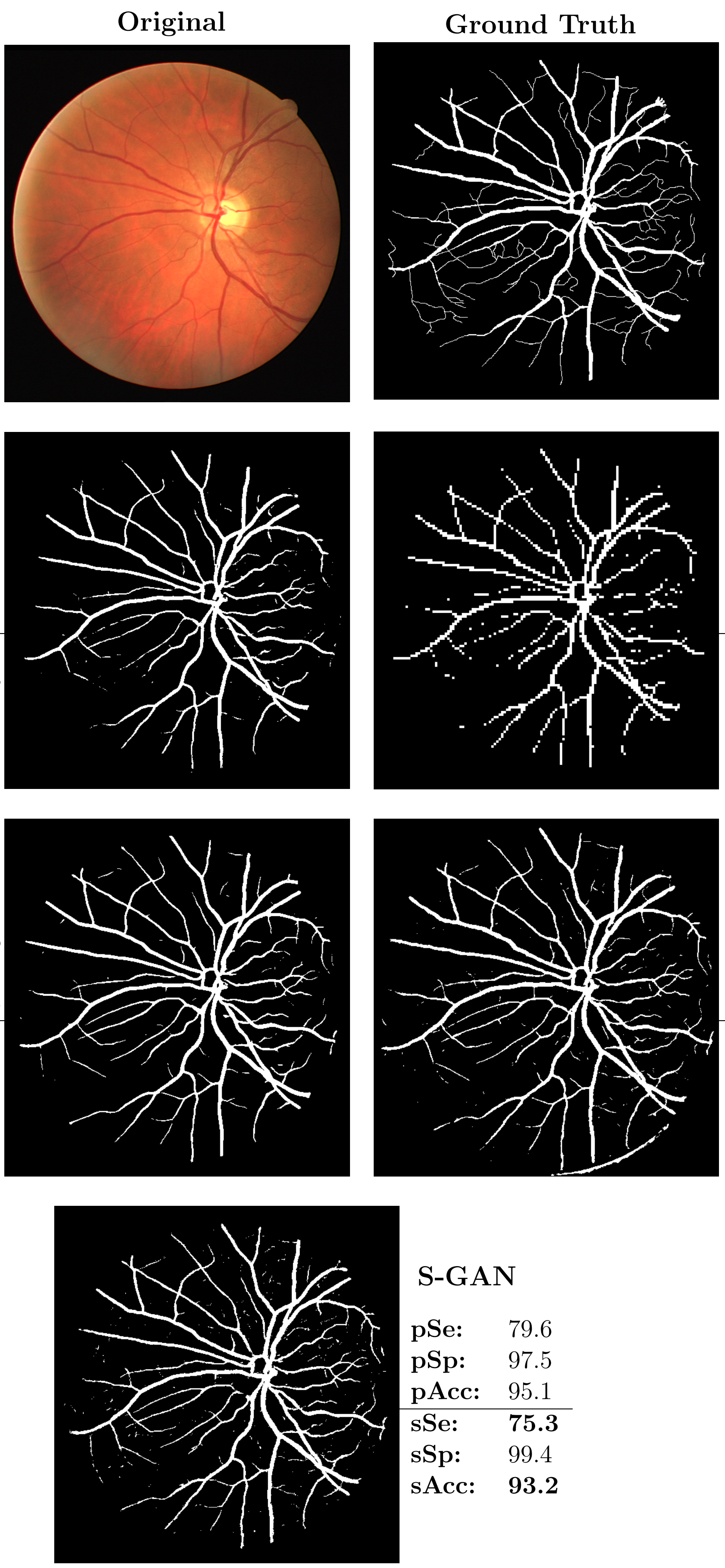
5.2 Results on DRIVE

The five methods listed in Table 5.1 were trained using the training images. For each epoch the model was evaluated in the validation set and, if the performance improved, the model was saved. Resulting images and metrics for one of the test images are shown in Fig. 5.5. Table 5.3 shows the average and standard deviation performance metrics of each of the five methods over the test set.

Table 5.3: *Average performance metrics on DRIVE test set.*

	Pixelwise			Structural		
Method	*pSe*	*pSp*	*pAcc*	*sSe*	*sSp*	*sAcc*
CNN	73.5 ±7.1	**98.0** ±0.8	94.9 ±0.5	65.6 ±8.5	99.6 ±0.2	90.9 ±2.4
FCN	60.6 ±7.8	96.9 ±0.9	92.2 ±0.6	41.6 ±7.5	98.8 ±0.3	84.2 ±2.6
UNet	77.8 ±7.0	97.8 ±0.7	**95.2** ±0.6	76.7 ±8.5	**99.6** ±0.2	93.7 ±2.4
SConvNet	75.6 ±7.9	97.8 ±0.7	94.9 ±0.7	75.7 ±8.4	99.4 ±0.3	93.3 ±2.3
S-GAN	**78.5** ±7.0	97.2 ±0.8	94.8 ±0.5	**78.9** ±8.3	99.2 ±0.3	**93.9** ±2.2

As it can be seen in Table 5.3, the standard deviation of the methods are very similar, and, with the exception of *Se*, the values are considerably low. Therefore, the average metrics are a good indicative of the overall performance of the algorithms.



CNN

pSe:	76.2
pSp:	98.2
pAcc:	95.3
sSe:	64.6
sSp:	**99.8**
sAcc:	90.7

FCN

pSe:	65.7
pSp:	**98.7**
pAcc:	92.6
sSe:	45.3
sSp:	98.7
sAcc:	85.0

UNet

pSe:	**80.6**
pSp:	97.8
pAcc:	**95.5**
sSe:	73.1
sSp:	99.7
sAcc:	92.8

SConvNet

pSe:	78.4
pSp:	97.9
pAcc:	95.3
sSe:	74.1
sSp:	99.6
sAcc:	93.0

S-GAN

pSe:	79.6
pSp:	97.5
pAcc:	95.1
sSe:	**75.3**
sSp:	99.4
sAcc:	**93.2**

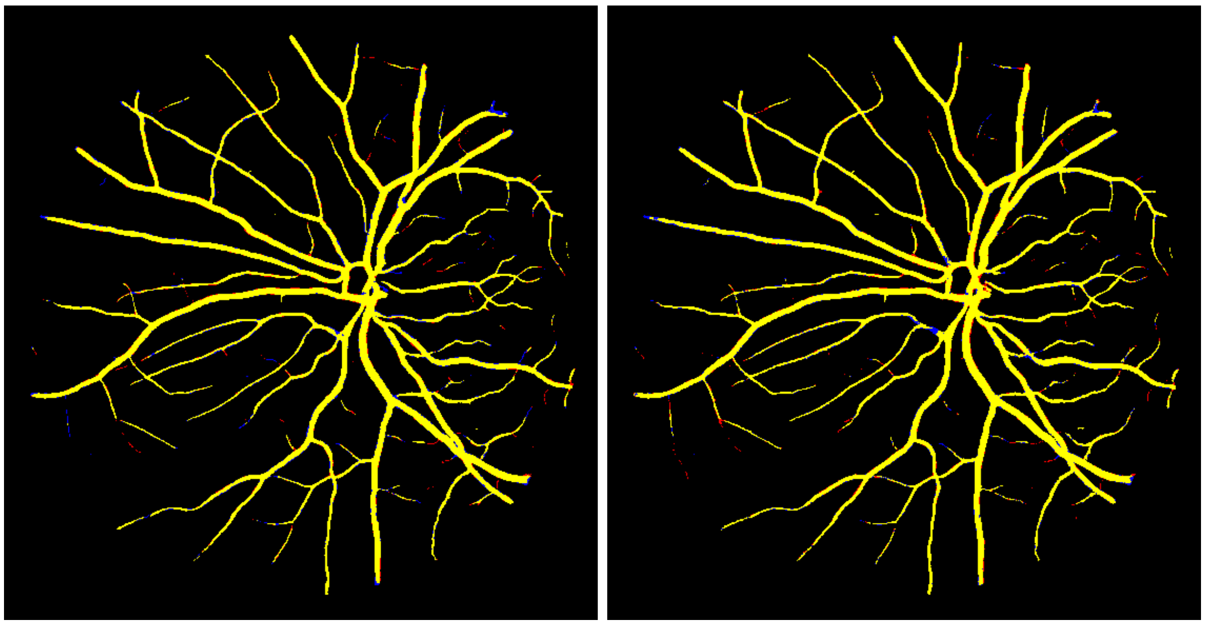
Figure 5.5: *Example of segmentation results by the five methods for a DRIVE test image.*

The first observation is regarding FCN. As expected, we can see it generates pixelated results as shown in Figure 5.5. This is also reflected in the average metrics shown in Table 5.3. Overall, only thick vessels are segmented and the edges of the vessels are not precisely defined. This is due to the upsampling technique used by FCN not accounting for the location information lost during the downsampling part. Sensitivity and accuracy, both pixelwise and structure-wise have the lowest scores with this method. FCN also has the largest difference between pixelwise and structural metrics, suggesting that part of the pixels found by FCN is not very meaningful to the overall structure.

As for CNN, it presents the highest specificity and a relatively low sensitivity. This can be explained by the fact that classes are highly unbalanced and no strategy to treat this condition was employed, therefore this method greatly optimizes the classification of background pixels. It is also interesting to note the difference between pixelwise and structural metrics. This difference in the performance of CNN is smaller than the difference in the metrics of FCN, suggesting that the pixels found by CNN are more meaningful to vessel structures than the ones found by the FCN method.

The remaining three methods, UNet, SConvNet and S-GAN, present more similar results each other, with UNet presenting a slightly better pSp and S-GAN presenting better pSe and sSe. It is interesting to compare the difference between pixelwise and structural metrics, as a higher pSe than sSe means that many pixels found are in the thickness part of the vessels, while a higher sSe means that the pixels found are more similar to the skeleton of the image. In the average performance metrics, S-GAN, SConvNet and UNet are the methods in which sSe and pSe are most similar, while CNN and FCN have a larger difference, suggesting that the pixels found by S-GAN, SConvNet and UNet have greater contributions to the overall structure of the output image than CNN and FCN.

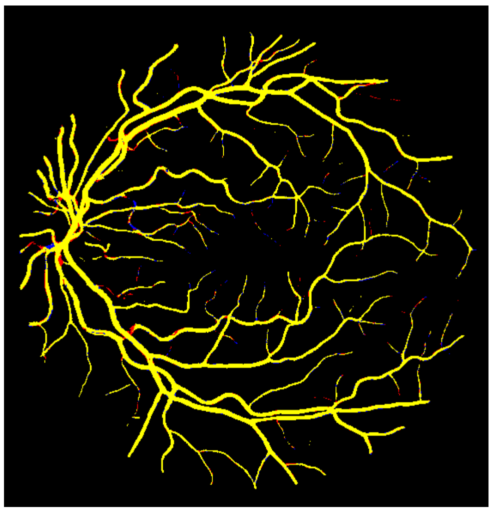
To further understand the differences among the three last methods, we computed a pairwise difference image between UNet and SConvNet (Fig. 5.6a) and between UNet and S-GAN (Fig. 5.6b) for the images shown in Figure 5.5. In both difference images, the yellow colour corresponds to vessel pixels found both by UNet and by the other method, while the blue colour corresponds to vessel pixels found only by UNet, and the red ones correspond to those found only by the other method. According to the metrics shown in Figure 5.5, for this test image UNet scores higher than both other methods in pSe and $pAcc$, but scores lower in their structural counterparts, suggesting that, while UNet finds more vessel pixels, SConvNet and S-GAN find more vessel structures. This can be seen in the difference images, in which most of the red pixels (vessel pixels only found by SConvNet and S-GAN) are located in thin vessels, indicating more thin structures found, and the blue ones (vessel pixels only found by UNet) are usually located alongside yellow pixels, indicating a better defined structure.

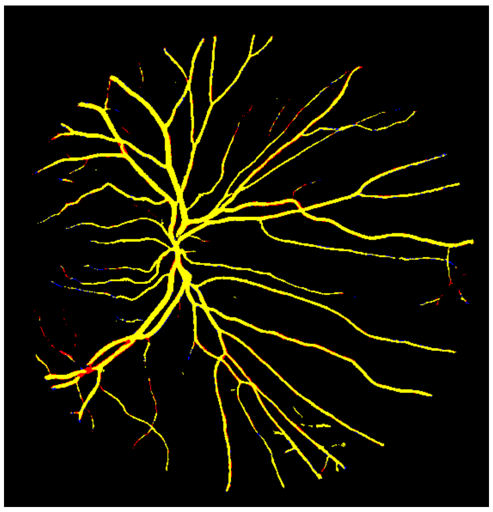


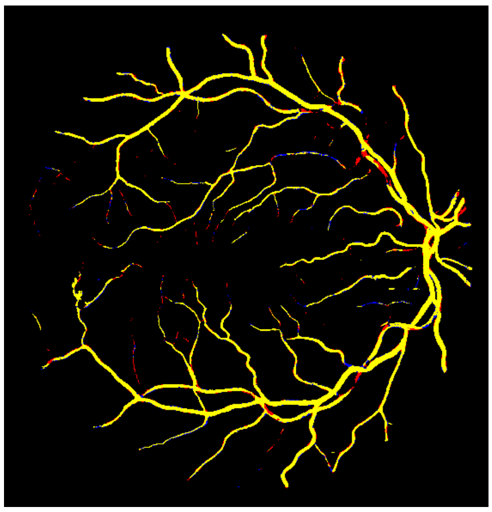
(a) *UNet × SConvNet* **(b)** *UNet × S-GAN*

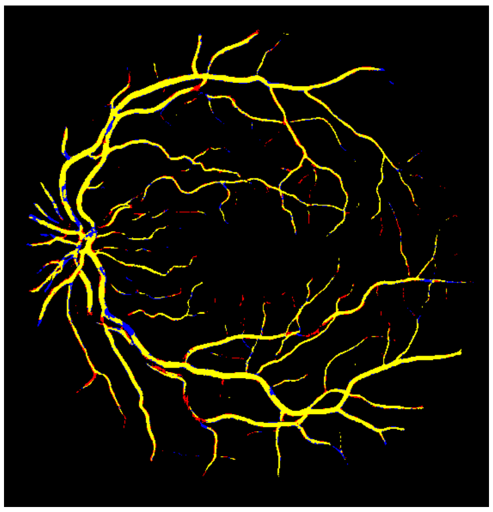
Figure 5.6: *Comparison images highlighting differences between the predictions by (a) UNet and SConvNet and by (b) UNet and S-GAN. Yellow represents vessel pixels found by both methods, blue represents vessel pixels found by UNet but missed by the other method, and red represents vessel pixels missed by UNet but found by the other method.*

Figure 5.7 shows additional results of S-GAN and UNet overlaid each other. An enlarged view of parts of these images are shown in Fig. 5.8. In these figures, yellow indicates pixels that have been correctly classified as positive by both, red indicates vessel pixels correctly found by S-GAN and missed by UNet and blue indicates vessel pixels correctly found by UNet and missed by S-GAN. Most of the thin vessels are red, indicating that S-GAN is better at finding those, as blue pixels are found in thicker vessels and usually alongside yellow ones, suggesting that UNet is better at defining the thickness of the vessels. This is also indicated in the average metrics, as S-GAN has a better structural accuracy and sensitivity, but UNet has a better overall pixelwise accuracy.


(a)


(b)


(c)


(d)

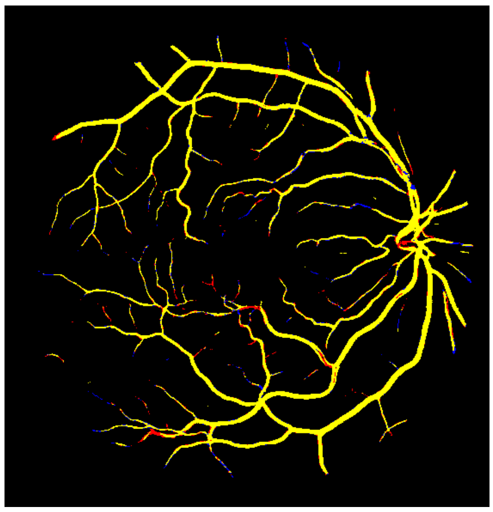

(e)

Figure 5.7: *Results of S-GAN and UNet for some of the DRIVE test images. Yellow indicates pixels found by both methods, red indicates pixels only found by S-GAN and blue indicates vessels that were only found by UNet.*

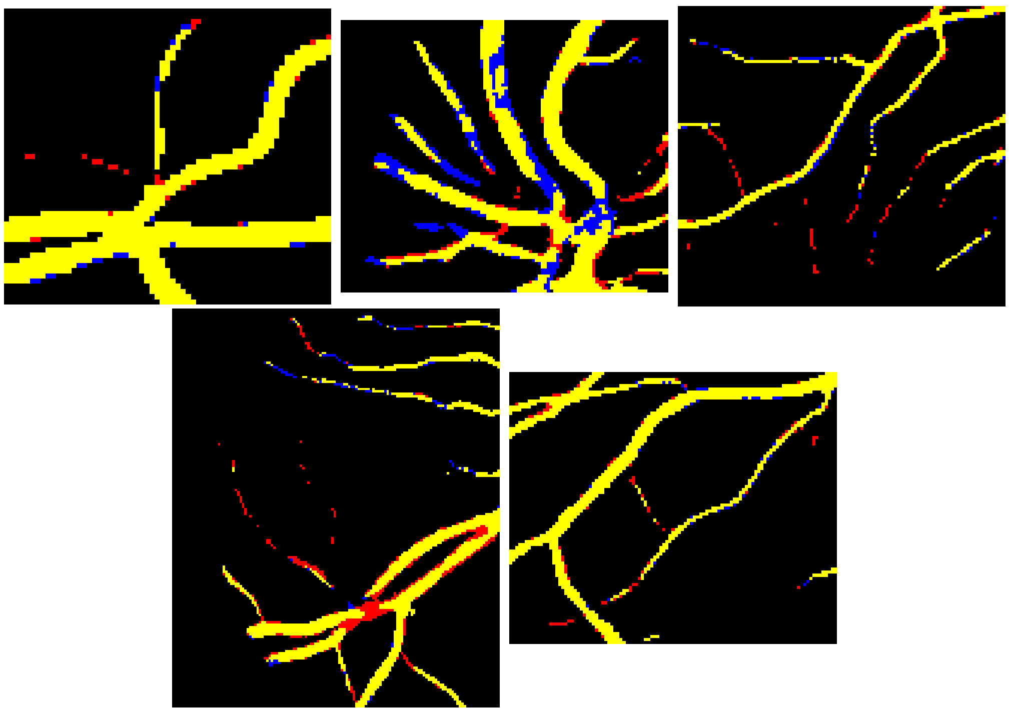


Figure 5.8: *Zoom in of patches from the comparison images of Fig. 5.7. Yellow indicates pixels found by both methods, red indicates pixels only found by S-GAN and blue indicates vessels that were only found by UNet.*

5.3 Results on DIBCO

Table 5.4 shows the average and standard deviation performance metrics of each of the six methods over the test set. Resulting images and metrics for one test image are shown in Fig. 5.9 (due to the large portions of background in the right and bottom borders of the image, we computed the metrics discarding that part).

Table 5.4: *Average performance metrics on DIBCO test set.*

	Pixelwise			Structural		
Method	*pSe*	*pSp*	*pAcc*	*sSe*	*sSp*	*sAcc*
CNN	86.1 ±13.6	91.4 ±5.2	91.1 ±4.4	92.2 ±10.8	91.4 ±5.2	91.6 ±4.7
FCN	70.1 ±13.4	91.3 ±7.8	89.4 ±6.9	73.7 ±17.5	91.7 ±8.0	90.0 ±7.0
UNet	**87.2** ±8.4	96.4 ±3.5	**95.5** ±3.3	**93.4** ±6.1	96.4 ±3.6	**96.0** ±3.3
SConvNet	80.1 ±22.2	96.3 ±3.6	94.6 ±4.0	87.4 ±18.6	96.3 ±3.7	95.1 ±3.9
S-GAN	68.6 ±26.1	95.3 ±5.2	92.7 ±4.9	77.1 ±26.8	95.4 ±5.3	93.2 ±5.3
U-GAN	63.4 ±29.7	**97.7** ±1.4	94.6 ±3.7	69.3 ±29.7	**98.0** ±1.3	**95.1** ±4.1

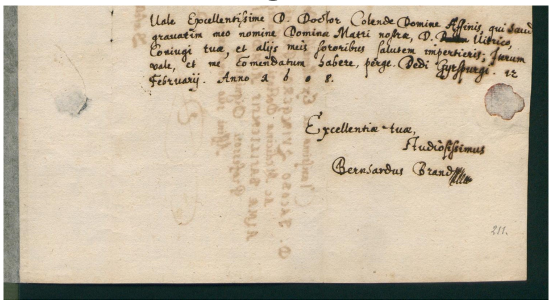
Differently than the observed on DRIVE dataset, FCN is not the worst performing method in the metrics (S-GAN and U-GAN scores lower in pixelwise sensitivity and U-GAN also scores lower structure-wise, but their overall accuracy is higher). However, by examining an example output image (illustrated in Figure 5.9), we can still see a highly pixelated output due to its inability to recover pixel location information.

UNet presents the best overall performance in the average metrics and also the lowest standard deviation. For the particular test image in Fig. 5.9, UNet also presents competitive values in all of the metrics. Pixelwise, the top three methods are UNet, followed by SConvNet and CNN. Among the last two, CNN presents better sensitivity while SConvNet presents better specificity.

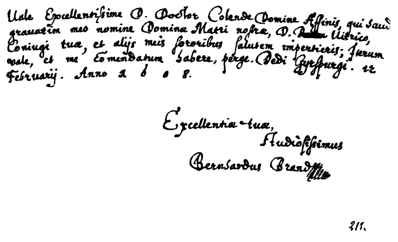
For the image in Fig. 5.9, CNN scores higher with respect to pixelwise evaluation metrics, however it is rated lower in structural metrics. Thus, although this method finds more foreground pixels, other methods such as S-GAN and SConvNet predict a more consistent structure compared to the ground truth. CNN's output also has more noise, indicated by the lower performance in specificity.

As for the GAN based methods, they perform better regarding specificity but relatively poorly regarding sensitivity. This means that in average they generate cleaner images (note for instance that it leaves less borders of the stain in the bottom right part of the image in Fig. 5.9), but at the same time they might miss a larger number of stroke pixels (see Fig. 5.10 for a poor performance). This is reflected in the high standard deviation, which shows that these methods can have good performance on specific situations,

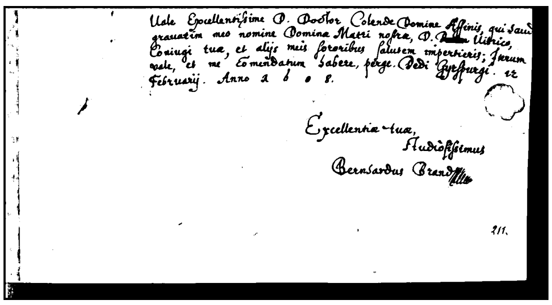
Original



Ground Truth

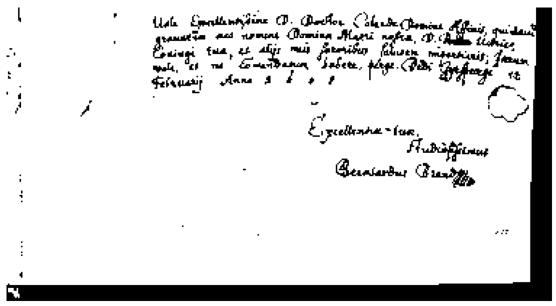


CNN



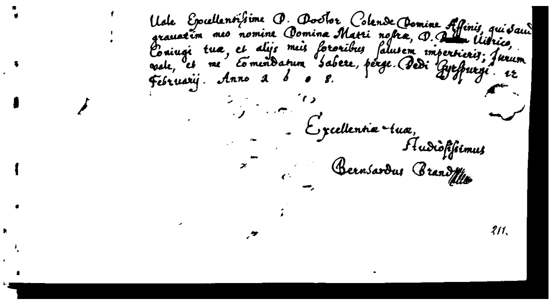
pSe	pSp	pAcc	sSe	sSp	sAcc
92.5	97.6	97.3	96.5	97.6	97.5

FCN



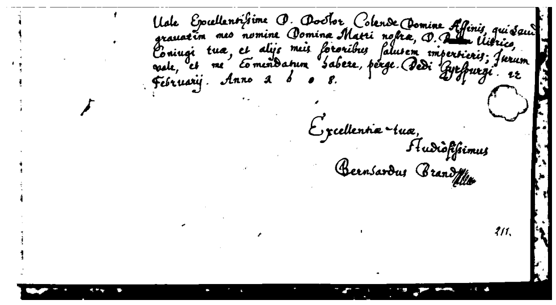
pSe	pSp	pAcc	sSe	sSp	sAcc
79.0	98.2	97.1	79.0	98.6	97.2

UNet



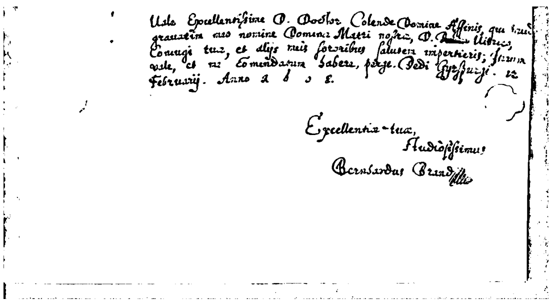
pSe	pSp	pAcc	sSe	sSp	sAcc
90.6	98.9	98.4	96.6	98.9	98.7

SConvNet



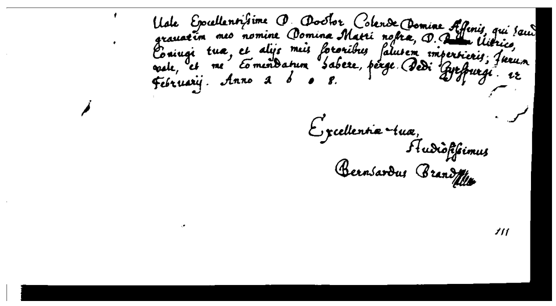
pSe	pSp	pAcc	sSe	sSp	sAcc
91.4	98.9	**98.5**	**97.5**	98.9	98.8

U-GAN



pSe	pSp	pAcc	sSe	sSp	sAcc
83.7	98.0	98.1	93.1	**99.1**	98.7

S-GAN



pSe	pSp	pAcc	sSe	sSp	sAcc
86.5	**99.1**	98.4	96.0	**99.1**	**98.9**

Figure 5.9: *Handwritten document binarized by different algorithms alongside pixelwise and structural metrics.*

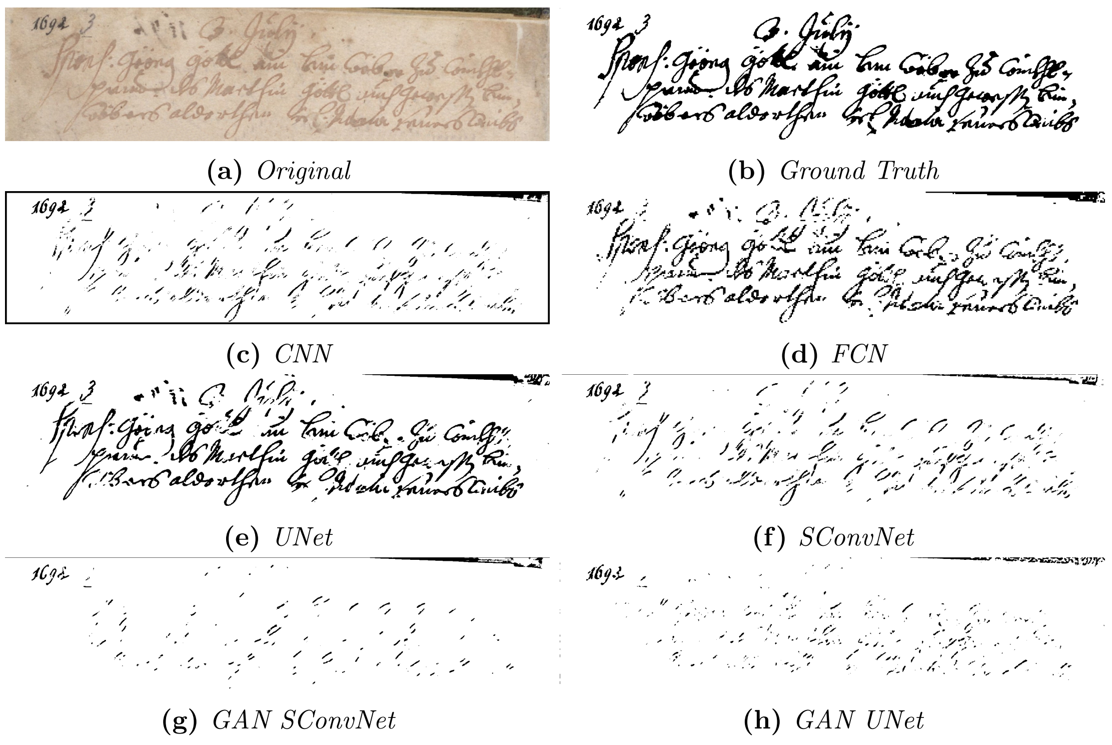

(a) *Original* **(b)** *Ground Truth* **(c)** *CNN* **(d)** *FCN* **(e)** *UNet* **(f)** *SConvNet* **(g)** *GAN SConvNet* **(h)** *GAN UNet*

Figure 5.10: *Image outputs from six image-to-image transformation learning methods, which shows the weakness of GANs with regard to classification of foreground pixels in low contrast images.*

but worse results in others. This poor performance could be caused by two factors. First, due to the large diversity of the dataset, it can be harder to model the structures of the strokes than the structures of vessels. Second, since both UNet and SConvNet present relatively higher sensitivity, this poor performance in sensitivity might be related to training issues (remember that the adversarial framework of GAN makes this method tricky and not as straightforward to train as the other ones). For the image in Fig. 5.9, both GANs have the largest differences between pixelwise and structural metrics, indicating that S-GAN and U-GAN training favors structural aspects. With some tweaks and an adjustment to the task at hand, these methods could possibly generate results that are similar to UNet and SConvNet in terms of sensitivity.

By analyzing the standard deviation of the methods (seen in Table 5.4), we can see that DIBCO is a more challenging dataset than DRIVE (all of the metrics have a higher deviation, with the exception of structural sensitivity of UNet). This is also a reflection of the diversity of the dataset, as in DIBCO we have, for instance, different stroke thickness and different contrasts between foreground and background. DRIVE dataset is more homogeneous with respect to these aspects.

5.4 Discussions

Both datasets share some similarities in the objects of interest, such as thin and elongated structures. These similarities are convenient for using the same evaluation metric in both, as shown in Chapter 4. However, during training some differences emerge. DIBCO is a more challenging dataset as stroke thickness is more varying than vessel thickness, there is a difference between writing styles, and there is a larger difference between the contrast of foreground and background between the images.

The difference of the dataset characteristics is reflected in the performance of the methods. Overall, all methods presented a more consistent performance on DRIVE images than on DIBCO images, as seen through visual inspection and by the computed metrics (smaller variance of the metrics among images in the DRIVE dataset in contrast to a larger variance on DIBCO images). Training of the models was also easier with respect to the DRIVE dataset, requiring less data and training time to converge.

Training the structural approaches was more difficult and required a few tricks to achieve convergence, such as label smoothing (training the algorithm with smooth labels, such as 0.9 and 0.1, instead of the common hard labels, such as 1 and 0) and different learning rates for Generator and Discriminator, while pixelwise and patch-to-patch training was more straightforward. Between the last two, pixelwise converged in a small number of epochs, but training each epoch required more time, while patch-to-patch needed more epochs to converge with less training time for each epoch. UNet, a patch-to-patch method, was more consistent throughout the datasets, being the one that presented smaller variance of the performance metrics on DIBCO.

Overall, structural approaches performed better on DRIVE, an easier task, where the variance of images is not large. In this task, favoring the structural aspects resulted in more thin vessels found. On DIBCO, a more complex task, where there is a larger variance among the strokes, results show that modeling the structural aspects of the objects and achieving better results is harder.

Chapter 6

Conclusion

The goal of this thesis was to compare image-to-image transformation learning approaches, investigating their respective advantages and drawbacks, while also giving an insight for other researchers into which approach should be suited for learning a desired image-to-image transformation. Therefore, we divided the approaches into three categories, separating them regarding their input-output pair relation during training. These three categories are pixelwise, patch-to-patch and structural. Then, we studied and implemented representative methods of each, to experimentally compare the approaches.

Comparison was based on the standard pixelwise metrics (sensitivity, specificity and accuracy) and also on structural similarity metrics. The structural similarity metrics were computed based on Skeletal Similarity counterparts of sensitivity, specificity and accuracy. These structural metrics were originally proposed for evaluating retinal vessel segmentation (Yan et al., 2018a). Therefore, we experimentally assessed their suitability for evaluating handwritten document image binarizations and showed that for this type of images the metrics are able to capture structural consistency between two images.

Results of the methods with respect to two datasets, DRIVE and DIBCO, suggested that pixelwise approach is best suited for tasks where the correct assignment of each pixel is the most important aspect of it. UNet method, a patch-to-patch approach, is one of the best performing methods throughout the experiments, indicating that it is one of the best methods in balancing performance of the transformation with the computational cost during prediction time. UNet was able to recover part of the information about pixel location that is lost during pooling layers. This suggests that UNet's concatenation between previous layers and upsampling layers is a reasonable strategy to recover such information. The experiments also suggested that structural approaches, such as GANs, are preferred in tasks where the most important aspect of the transformed image is an overall structural consistency, although training these methods is harder than training the other methods. In fact, the GAN based methods were able to model the structures in DRIVE images, a relatively homogeneous dataset, finding thin vessels not found by other methods. However, they did not perform well on DIBCO, which is a more heterogeneous and challenging dataset.

By investigating the three approaches, we came up with a technique called SConvNet. This technique is a theoretical combination of pixelwise and patch-to-patch approaches, which joins the advantages of both without their main drawbacks. Thus, it is a technique that can be trained pixel-by-pixel and it can be applied in the whole image at once. SConvNet had competitive results in the DRIVE dataset but it showed a worst result in DIBCO, a more challenging dataset.

Hence, the contributions of this thesis can be summarized as the following items:

- Discussion and description of main state of the art methods for the image-to-image transformation learning problem, and an experimental comparison between representative methods of each.
- Development of SConvNet, a method that unites the advantages of two approaches, pixelwise and patch-to-patch. This network is also simpler than most fully convolutional networks, thus it can be used as an initial approach when tackling an image-to-image transformation learning problem.
- Investigation of the use of Skeletal Similarity metrics to evaluate the performance of handwriting document binarization algorithms. This metric better captures the structural consistency between two document images than pixelwise metrics. Thus, it can be used in a complementary way for the evaluation of document binarization algorithms.

6.1 Future Works

Most recent works in binarization use pixelwise Precision and Recall as performance metrics. We confirmed that Skeletal Similarity metrics are useful to measure the structural consistency between image transformations. However, while Recall is equivalent to Sensitivity, having therefore a Skeletal Similarity counterpart, there is still no direct counterpart of the Precision metric for Skeletal Similarity. Thus, a structural Precision metric should be developed and experimented upon, to structurally compare these metrics in state-of-the-art works.

Another direction to be researched is to use the Skeletal Similarity metrics as loss function in the training step of some algorithms. This idea was proposed in Yan et al. (2018b), where the thickness part of the Skeletal Similarity algorithm was used in conjunction with the pixelwise loss to improve upon previous results, however it did not use the curve similarity part of the Skeletal Similarity algorithm. Adapting such metrics to be used as loss functions could improve the structural consistency of machine learning algorithms.

Appendix A

Binarization Methods

A.1 Otsu

Otsu's algorithm for binarization of images [Otsu (1979)] returns an intensity threshold value to separate pixels into foreground and background that minimizes intra-class variance. Intra-class variance is defined as:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{A.1}$$

where weights ω_0 and ω_1 are the estimated probabilities of each class when threshold t is employed (which is calculated from the image histogram) and σ_0^2 and σ_1^2 are the sample variance of each class. For 2 class problems, minimizing intra-class variance is equivalent to maximizing inter-class variance, as:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) \\ &= \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_T(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned} \tag{A.2}$$

where $\mu_0(t)$ and $\mu_1(t)$ are the mean values of each class.

Then, Otsu's method computes $\sigma_b^2(t)$ for each possible value for t, and returns the one that corresponds to the maximum inter-class variance $\sigma_b^2(t)$.

A.2 Sauvola

An overview of Sauvola's algorithm [Sauvola and Pietikäinen (2000)] can be seen in Figure A.1.

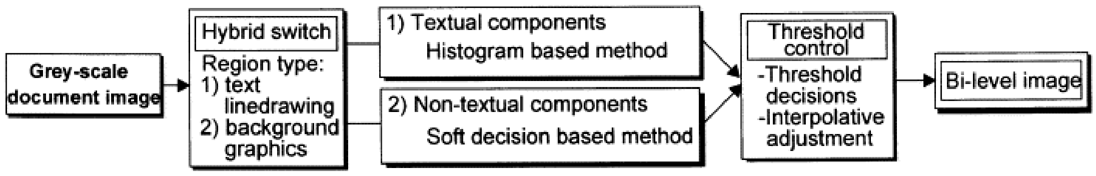


Figure A.1: *Overview of Sauvola's binarization method (source: Sauvola and Pietikäinen (2000)).*

The algorithm aims to find a threshold value for each pixel, instead of a global one like Otsu's method. The image is divided into tiles, and each tile passes through the algorithm displayed in Figure A.1. First, according to extracted features of the tile, one of two binarization algorithms, either one specific for textual components or one specific for non-textual, is chosen. Then, the chosen algorithm is applied to the tile returning a threshold value for each pixel of the tile. For non-textual components, a Soft Decision Method (SDM), which includes noise filtering is used, while for text-components a specialized text binarization method, such as Niblack's method from Niblack (1985), is used. The results of each tile are then combined to return the binarized image.

A.3 Howe

Howe's method [Howe (2011)] advanced the state-of-the-art in document binarization field at the time. Even nowadays is considered one of the best performing methods in document binarization. For instance, the best evaluated method in DIBCO 2018, after a pre-processing step, use this method for binarization of the images.

This method is based on three main strategies. First, binarization of a document image is considered as a pixel labeling that minimizes a global energy function (a function that takes a typical additive form, with terms representing data fidelity and smoothness/regularity of the binarization), inspired by Markov Random Field models. Second, the data fidelity term of this function relies on the Laplacian of the image intensity (which reduces the problem of light or intensity variance across the image). Third, the smoothness term of this energy function includes edge discontinuities, encouraging the edges to match stroke boundaries and stronger smoothness over the rest of the image. This global energy function is defined as:

$$\begin{aligned} E = &\sum_{i=0}^{m}\sum_{j=0}^{n}[L_{ij}^{0}(1-B_{ij}) + L_{ij}^{1}B_{ij}] \\ &+ \sum_{i=0}^{m}\sum_{j=0}^{n} C_{ij}^{h}(B_{ij} \neq B_{i+1,j}) \\ &+ \sum_{i=0}^{m}\sum_{j=0}^{n} C_{ij}^{v}(B_{ij} \neq B_{i,j+1}) \end{aligned} \tag{A.3}$$

where the first term is the data fidelity term (L_{ij}^0 and L_{ij}^1 are the costs of assigning label B_{ij} to pixel ij), and the last two are the smoothness terms (C_{ij}^v and C_{ij}^h are the costs of assigning a different value to B_{ij} than to its neighbor below or to the left, respectively). Costs L_{ij}^0 and L_{ij}^1 are taken from the Laplacian image intensity and costs C_{ij}^h and C_{ij}^v are set to a constant c except where a Canny edge filter (Canny (1986)) identifies a likely discontinuity.

This global energy function can be solved by maximum flow methods, which can efficiently compute the optimal binarization (Boykov and Kolmogorov (2004)). Parameters of this method can be automatically tuned, as shown in (Howe (2013)).

A.4 Zaragosa

Differently from the previously described algorithms, Zaragosa's method uses machine learning technique to learn an end-to-end mapping of a document image to a binary version of it (Calvo-Zaragoza and Gallego (2019)). This method trains an auto-encoder, called by the authors as *Selectional Auto-Encoder* (SAE), by feeding the network with document images and the correspondening binarized ground truth.

This architecture can be divided into two parts, an encoding part that consists of a sequence of convolutions and downsampling layers, and a decoder part, which consists of a sequence of convolutions and upsampling layers. An overview of it can be seen in Figure A.2.

After trained, this architecture can binarize a document image with only one forward pass.

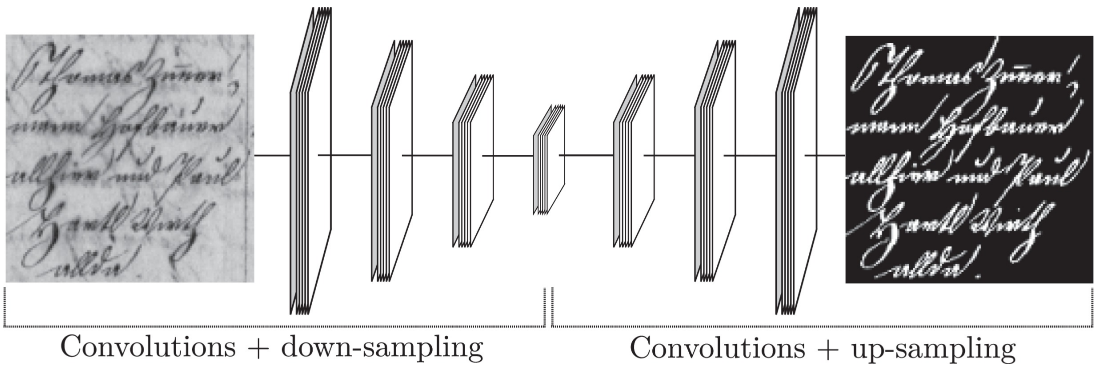


Figure A.2: *Overview of a SAE used for document binarization (source: Calvo-Zaragoza and Gallego (2019)).*

Bibliography

Arnab et al.(2016) A. Arnab, S. Jayasumana, S. Zheng and P. H. S. Torr. Higher order conditional random fields in deep neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe and Max Welling, editors, Computer Vision – ECCV 2016, pages 524–540, Cham. Springer International Publishing. ISBN 978-3-319-46475-6. Cited on page. 17

Badrinarayanan et al.(2017) V. Badrinarayanan, A. Kendall and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12):2481–2495. doi: 10.1109/TPAMI.2016.2644615. Cited on page. 2, 20

Barrera et al.(1997) J. Barrera, E. R. Dougherty and N. S. Tomita. Automatic programming of binary morphological machines by design of statistically optimal operators in the context of computational learning theory. Journal of Electronic Imaging, 6(1): 54–67. doi: 10.1117/12.260010. Cited on page. 2

Boykov and Kolmogorov(2004) Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9):1124–1137. Cited on page. 67

Brock et al.(2019) A. Brock, J. Donahue and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In International Conference on Learning Representations. URL https://openreview.net/forum?id=B1xsqj09Fm. Cited on page. 15

Buades et al.(2005) A. Buades, B. Coll and J. . Morel. A non-local algorithm for image denoising. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 60–65 vol. 2. Cited on page. 1

Calvo-Zaragoza and Gallego(2019) J. Calvo-Zaragoza and A. Gallego. A selectional auto-encoder approach for document image binarization. Pattern Recognition, 86:37 – 47. doi: https://doi.org/10.1016/j.patcog.2018.08.011. URL http://www.sciencedirect.com/science/article/pii/S0031320318303091. Cited on page. xi, 34, 37, 67, 68

Calvo-Zaragoza et al.(2017) J. Calvo-Zaragoza, A. Pertusa and J. Oncina. Staff-line detection and removal using a convolutional neural network. Machine Vision and Applications, 28(5-6):665–674. Cited on page. 10

Canny(1986) J. Canny. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6):679–698. Cited on page. 1, 67

Chen et al.(2015) L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In Proc. Int. Conf. Learning Representations. URL http://arxiv.org/abs/1412.7062. Cited on page. ix, 16

Chen et al.(2009) T. Chen, M. Cheng, P. Tan, A. Shamir and S. Hu. Sketch2Photo: Internet Image Montage. ACM Trans. Graph., 28(5):1–10. doi: 10.1145/1618452.1618470. URL https://doi.org/10.1145/1618452.1618470. Cited on page. 1

Cireşan et al.(2012) D. C. Cireşan, A. Giusti, L. M. Gambardella and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In International Conference on Neural Information Processing Systems, NIPS'12, pages 2843–2851, USA. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999325.2999452. Cited on page. 10

Dellamonica Jr. et al.(2007) D. Dellamonica Jr., P. J. S. Silva, C. Humes Jr., N. S. T. Hirata and J. Barrera. An exact algorithm for optimal MAE stack filter design. IEEE Transactions on Image Processing, 16(2):453–462. Cited on page. 10

Fergus et al.(2006) R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis and W. T. Freeman. Removing camera shake from a single photograph. In ACM SIGGRAPH 2006 Papers, SIGGRAPH '06, page 787–794, New York, NY, USA. Association for Computing Machinery. doi: 10.1145/1179352.1141956. Cited on page. 1

Goodfellow et al.(2014) I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative Adversarial Nets. In Z Ghahramani, M Welling, C Cortes, N D Lawrence and K Q Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc. Cited on page. 18

Goodfellow et al.(2016) I. Goodfellow, Y. Bengio and A. Courville. Deep Learning. MIT Press. http://www.deeplearningbook.org. Cited on page. 10

Haiping Lu et al.(2004) Haiping Lu, A. C. Kot and Y. Q. Shi. Distance-reciprocal distortion measure for binary document images. IEEE Signal Processing Letters, 11(2): 228–231. doi: 10.1109/LSP.2003.821748. Cited on page. 33

Hedjam and Cheriet(2013a) R. Hedjam and M. Cheriet. Ground-truth estimation in multispectral representation space: Application to degraded document image binarization. In 12th International Conference on Document Analysis and Recognition, pages 190–194. doi: 10.1109/ICDAR.2013.45. Cited on page. 35

Hedjam and Cheriet(2013b) R. Hedjam and M. Cheriet. Historical document image restoration using multispectral imaging system. Pattern Recognition, 46(8):2297–2312. doi: 10.1016/j.patcog.2012.12.015. Cited on page. 35

Heijmans(1994) H. J. A. M. Heijmans. Morphological Image Operators. Academic Press, Boston. Cited on page. 9

Howe(2013) N. R. Howe. Document binarization with automatic parameter tuning. International Journal on Document Analysis and Recognition, 13(3):247–258. DOI: 10.1007/s10032-012-0192-x. Cited on page. 67

Howe(2011) N. R. Howe. A laplacian energy for document binarization. In International Conference on Document Analysis and Recognition, pages 6–10. Cited on page. 37, 66

Isola et al.(2017) P. Isola, J. Zhu, T. Zhou and A. A. Efros. Image-to-image translation with conditional adversarial networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5967–5976. doi: 10.1109/CVPR.2017.632. Cited on page. ix, 2, 13, 15, 19

Julca-Aguilar and Hirata(2017) F. D. Julca-Aguilar and N. S. T. Hirata. Image operator learning coupled with CNN classification and its application to staff line removal. In 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pages 53–58, Los Alamitos, CA, USA. IEEE Computer Society. doi: 10.1109/ICDAR.2017.18. Cited on page. 2, 8, 10

Karras et al.(2018) T. Karras, T. Aila, S. Laine and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In International Conference on Learning Representations. URL https://openreview.net/forum?id=Hk99zCeAb. Cited on page. 15

Krizhevsky et al.(2012) A. Krizhevsky, I. Sutskever and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, USA. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257. Cited on page. 11

Lafferty et al.(2001) J. D. Lafferty, A. McCallum and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Cited on page. 15

Li and Wand(2016) C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In Bastian Leibe, Jiri Matas, Nicu Sebe and Max Welling, editors, Computer Vision – ECCV 2016, pages 702–716, Cham. Springer International Publishing. ISBN 978-3-319-46487-9. Cited on page. 19

Li et al.(2016) M. Li, A. Shekhovtsov and D. Huber. Complexity of discrete energy minimization problems. In Bastian Leibe, Jiri Matas, Nicu Sebe and Max Welling, editors, Computer Vision – ECCV 2016, pages 834–852. Springer International Publishing. Cited on page. 16

Liu and Deng(2015) S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pages 730–734. doi: 10.1109/ACPR.2015.7486599. Cited on page. 11

Long et al.(2015) J. Long, E. Shelhamer and T. Darrell. Fully convolutional networks for semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440. doi: 10.1109/CVPR.2015.7298965. Cited on page. ix, 2, 3, 12, 13, 20

Melinscak et al.(2015) M. Melinscak, P. Prentasic and S. Loncaric. Retinal vessel segmentation using deep neural networks. In Proceedings of the 10th International Conference on Computer Vision Theory and Applications, pages 577–582. INSTICC, SciTePress. doi: 10.5220/0005313005770582. Cited on page. 10

Mirza and Osindero(2014) M. Mirza and S. Osindero. Conditional generative adversarial nets. CoRR, abs/1411.1784. URL http://arxiv.org/abs/1411.1784. Cited on page. 18

Montagner et al.(2016) I. S. Montagner, Nina S. T. Hirata and R. Hirata Jr. Image operator learning and applications. In 29th Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T). Cited on page. 2, 9

Montagner et al.(2017) I. S. Montagner, N. S. T. Hirata and R. Hirata Jr. Staff removal using image operator learning. Pattern Recognition, 63:310 – 320. doi: http://dx.doi.org/10.1016/j.patcog.2016.10.002. Cited on page. 8, 10

Niblack(1985) W. Niblack. An Introduction to Digital Image Processing. Strandberg Publishing Company, DNK. Cited on page. 66

Ntirogiannis et al.(2013) K. Ntirogiannis, B. Gatos and I. Pratikakis. Performance evaluation methodology for historical document image binarization. IEEE Transactions on Image Processing, 22(2):595–609. doi: 10.1109/TIP.2012.2219550. Cited on page. 33

Otsu(1979) N. Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1):62–66. doi: 10.1109/TSMC.1979.4310076. Cited on page. 37, 65

Pathak et al.(2016) D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell and A. A. Efros. Context encoders: Feature learning by inpainting. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2536–2544. Cited on page. 19

Pizer et al.(1987) S. M. Pizer, E. P. Amburn, J.n D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman and K. Zuiderveld. Adaptive histogram equalization and its variations. Computer Vision, Graphics, and Image Processing, 39(3):355 – 368. doi: https://doi.org/10.1016/S0734-189X(87)80186-X. URL http://www.sciencedirect.com/science/article/pii/S0734189X8780186X. Cited on page. 1

Pratikakis et al.(2018) I. Pratikakis, K. Zagoris, P. Kaddas and B. Gatos. ICFHR 2018 competition on handwritten document image binarization (H-DIBCO). In 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 489–493. doi: 10.1109/ICFHR-2018.2018.00091. Cited on page. x, 33, 37

Ronneberger et al.(2015) O. Ronneberger, P. Fischer and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI, pages 234–241. Springer International Publishing. Cited on page. ix, 2, 12, 13, 14, 20

Sauvola and Pietikäinen(2000) J. Sauvola and M. Pietikäinen. Adaptive document image binarization. Pattern Recognition, 33(2):225 – 236. doi: https://doi.org/10.1016/S0031-3203(99)00055-2. URL http://www.sciencedirect.com/science/article/pii/S0031320399000552. Cited on page. xi, 37, 66

Silva et al.(2020) A. C. M. Silva, N. S. T. Hirata and X. Jiang. Skeletal similarity based structural performance evaluation for document binarization. In 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 37–42. doi: 10.1109/ICFHR2020.2020.00018. Cited on page. 4

Staal et al.(2004) J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging, 23(4):501–509. Cited on page. 45

Suzuki and Keiichi(1987) S. Suzuki and A. Keiichi. Binary picture thinning by an iterative parallel two-subcycle operation. Pattern Recognition, 20(3):297–307. doi: 10.1016/0031-3203(87)90005-7. URL https://doi.org/10.1016/0031-3203(87)90005-7. Cited on page. 26

Szegedy et al.(2015) C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. In Computer Vision and Pattern Recognition (CVPR). URL http://arxiv.org/abs/1409.4842. Cited on page. 11

Yan et al.(2018a) Z. Yan, X. Yang and K. Cheng. A skeletal similarity metric for quality evaluation of retinal vessel segmentation. IEEE Transactions on Medical Imaging, 37 (4):1045–1057. doi: 10.1109/TMI.2017.2778748. Cited on page. x, 23, 24, 26, 27, 61

Yan et al.(2018b) Z. Yan, X. Yang and K. T. Cheng. Joint segment-level and pixel-wise losses for deep learning based retinal vessel segmentation. IEEE Transactions on Biomedical Engineering, 65(9):1912–1923. doi: 10.1109/TBME.2018.2828137. Cited on page. 63

Yu and Koltun(2016) Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun, editors, 4th International Conference on Learning Representations, ICLR. URL http://arxiv.org/abs/1511.07122. Cited on page. 22

Zhang et al.(2016) R. Zhang, P. Isola and A. A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe and Max Welling, editors, Computer Vision – ECCV 2016, pages 649–666, Cham. Springer International Publishing. Cited on page. 1

Zheng et al.(2015) S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang and P. H. S. Torr. Conditional random fields as recurrent neural networks. In IEEE International Conference on Computer Vision (ICCV), pages 1529–1537. doi: 10.1109/ICCV.2015.179. Cited on page. 2, 17