Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2019-04-01

Deep Learning for Document Image Analysis

Christopher Alan Tensmeyer
Brigham Young University

Deep Learning for Document Image Analysis

Christopher Alan Tensmeyer

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Tony R. Martinez, Chair
Ryan Matthew Farrell
Bryan Stuart Morse
Sean C. Warnick
Parris K. Egbert

Department of Computer Science

Brigham Young University

ABSTRACT

Deep Learning for Document Image Analysis

Christopher Alan Tensmeyer
Department of Computer Science, BYU
Doctor of Philosophy

Automatic machine understanding of documents from image inputs enables many applications in modern document workflows, digital archives of historical documents, and general machine intelligence, among others. Together, the techniques for understanding document images comprise the field of Document Image Analysis (DIA). Within DIA, the research community has identified several sub-problems, such as page segmentation and Optical Character Recognition (OCR). As the field has matured, there has been a trend of moving away from heuristic-based methods, designed for particular tasks and domains of documents, and moving towards machine learning methods that learn to solve tasks from examples of input/output pairs. Within machine learning, a particular class of models, known as deep learning models, have established themselves as the state-of-the-art for many image-based applications, including DIA. While traditional machine learning models typically operate on features designed by researchers, deep learning models are able to learn task-specific features directly from raw pixel inputs.

This dissertation is collection of papers that proposes several deep learning models to solve a variety of tasks within DIA. The first task is historical document binarization, where an input image of a degraded historical document is converted to a bi-tonal image to separate foreground text from background regions. The next part of the dissertation considers document segmentation problems, including identifying the boundary between the document page and its background, as well as segmenting an image of a data table into rows, columns, and cells. Finally, a variety of deep models are proposed to solve recognition tasks. These tasks include whole document image classification, identifying the font of a given piece of text, and transcribing handwritten text in low-resource languages.

Keywords: document image analysis, deep learning, binarization, document classification, font recognition, handwriting recognition

ACKNOWLEDGMENTS

I would like to begin by thanking those that have made my graduate work financially possible. Ancestry.com provided generous support and an interesting problem that sparked my interest in document analysis research. Adobe as well funded me through a research internship. Also, many thanks to the many faithful members of The Church of Jesus Christ of Latter-day Saints, for without their faithful sacrifice and donations, this wonderful university might not exist.

I thank my advisor, Tony Martinez, for supporting me, for prompt feedback, and for giving me the opportunity to pursue and develop my own research ideas. I've learned so much in this process. I also thank my other professors that have impacted not only my research direction, but also my research philosophy. Among others, I thank Bill Barrett, Ryan Farrell, Bryan Morse, Eric Ringger, and Sean Warnick.

Lastly, I thank my family, especially my loving and supportive wife, Nicole. Though it was a sacrifice for her, she has always encouraged me to pursue this academic goal, even if that meant greatly extending our time as "starving students." Finally, I thank my two sons, Kaden and Rowan, for the joy they are in my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This body of work fits into the field of Document Image Analysis (DIA), which can be broadly characterized as a large group of techniques for understanding documents from visual information. DIA is among the oldest fields of computer science, pre-dating even the first programmable computer. In 1913, Edmund Fournier invented the optophone which could detect dark character strokes on a piece of paper and produce tones that, with much effort, enabled the blind to interpret documents without resorting to braille [3]. While the optophone relied on the conductivity properties of selenium, contemporary DIA algorithms run on the silicon chips found in modern computing hardware and operate on raster images produced with scanners, cameras, and other digitization devices.

1.1 Tasks

Some of the earliest applications of DIA involved the recognition of printed characters, commonly known as Optical Character Recognition (OCR) [4]. Such technology is now widely deployed for parsing zip codes on mailed envelops to automate mail routing. One might claim that for noise-free, machine printed, scanned office documents in English, OCR is a solved problem. Indeed, some systems show higher than 99% accuracy on such cases [1]. However, there are over 3000 written languages [2], and OCR for the vast majority of these languages remains unsolved, though improving OCR accuracy for common languages is the subject of on-going research. Another common use of DIA technology is in form processing. When a person fills out a paper form (e.g. in a doctors office), it saves human effort if, after scanning the form to produce a digital image, the form can be automatically converted into key and value pairs and stored in a database. Such a process is comprised of a number of steps such as removing scanner noise, identifying sets of field labels and field values, and finally recognizing the text. All of these steps are challenging and much research effort has been expended to solve these problems.

As the field has matured, the variety of problems tackled has grown. The following is a non-exhaustive list of common problems encountered in the DIA literature.

- Binarization - Classifying which pixels are foreground and which are background.
- Rectification - Correcting for perspective distortion in camera captured document images. This occurs when the camera is not positioned orthogonally w.r.t. the physical document.
- Page Segmentation - Dividing a page into homogenous components such as text, images, and graphics.
- Text Line Segmentation - Dividing a paragraph of text into lines. This is difficult for handwritten text where acenders of one line overlap the descenders of the line above.
- Document Classification - Classifying the type of a document. E.g. form, memo, email, advertisement.
- Font Recognition - Classifying which font was used to render a given image of text.
- Language/Script Recognition - Classifying the language and/or script (writing system; e.g. Latin or Cyrillic) of text.
- Character Segmentation - Segmenting a word image into individual characters. Often required for OCR.
- Graphics Recognition - Interpretation of graphical components of documents, such as engineering drawings.

- Table Layout Decomposition - Recovering the row and column structure from an image of a table.
- Text in the Wild Detection and Recognition - Detection and recognizing text in natural scenes, such as recognizing the text of street signs or billboards.
- Writer Identification - Determining the individual or characteristics (e.g. gender, handedness) of the individual who wrote a certain piece of text.
- Document Image Quality Assessment - Determining the quality (e.g. legibility, blurriness) of a document image.
- Handwriting Recognition - Similar to OCR, but for handwritten text. Often, whole words are recognized at once because it can be difficult to segment a cursive word into characters.

There are many more topics that are the subject of active research.

The typical document image processing pipeline is composed of 3 generic steps.

1. Pre-processing - Noise removal, blur removal, rectification, deskewing, binarization
2. Layout Analysis - Understanding document structure to, e.g., identify Regions of Interest (RoI)
3. Recognition - Extracting application-specific information from each RoI

In general, pre-processing and layout analysis steps depend on the types and structure of documents under consideration. For example, segmenting form images is different from segmenting text lines in a handwritten manuscript. The recognition step is dependent on the end application. Commonly, applications aim to transcribe the textual content on the page to, e.g., create a searchable index or populate a database.

In addition to the growing number of problems found in the literature, there has been an expansion in the domains of documents considered. While much early research was focused on automating paper workflows in modern business settings, there is now a high interest in processing historical documents. DIA for historical documents is challenging due to the high levels of noise arising from the age and degradation of the documents. For example, recognition rates for historical text is much lower than that of modern office documents. Many historical archives have made digital copies of their documents available online, but without searchable digital transcriptions of the documents, it is difficult for users to find documents relevant to their needs. Manual transcription is costly, so automatically transcribing historical records to create searchable indexes with DIA techniques is a preferred solution if high enough accuracy can be obtained.

1.2 Techniques

With the growing variety of tasks and domains has come a shift in the fundamental techniques used in DIA. Early applications relied heavily on heuristic rules that applied to narrow document domains, but did not generalize well to all classes of documents. Supervised machine learning techniques, where a generalizable model is learned from human-annotated examples of input and output pairs, are now commonly used. Instead of inventing new heuristic rules for every language/domain/task combination, the same machine learning model can be applied to different sets of labeled data.

While traditional machine learning models operate over task-specific features chosen by the user, a certain class of hierarchical models, called deep neural networks, learn superior task-specific features directly from image pixels. Deep learning has recently transformed many research fields, including computer vision, natural language processing, and bioinformatics, and DIA is no exception. Each layer in the deep network abstracts the representation learned by the previous layer, gradually transforming an input image into high level semantic information, such as the type of document.

Due to the success of deep learning models in computer vision, many deep models designed for recognition of natural images have been ported directly to DIA tasks. In general, such attempts have been widely successful and outperform previous machine learning approaches that use hand-crafted features. However, given that natural images and document images have significant differences, it is likely that this approach to DIA has its limitations. This dissertation contains several deep learning approaches proposed for a variety of DIA tasks. In each case, the deep learning model is adapted for the DIA task under consideration and outperforms the naive application of a deep model transfered from the domain of natural images.

1.3 Summary of Contributions

Each chapter in this dissertaton, besides the introduction and concluding chapter, is a paper that has been published or submitted for peer-review. These papers have been organized into 3 parts on the topics of binarization, segmentation and recognition.

Part I is on binarization and consists of Chapters 2-4. The literature review in Chapter 2 is the first of its kind and provides a roadmap for future research in historical document binarization. Chapter 3 introduces a new multi-scale FCN model appropriate for binarization tasks. This model is able to learn features at multiple scales, including full resolution so that it can properly localize the boundary between foreground text and background. This FCN is trained using the proposed pseudo F-measure loss, which weights each pixel based on its location relative to the ground truth foreground regions so that errors are less likely to occur on more important pixels. This FCN model achieves state-of-the-art performance on several of the Document Image Binarization Competition (DIBCO) datasets. Chapter 4 extends this line of work to construct realistic ground truth so that deep learning models in binarization can be trained with sufficient data. Because explicit modeling of realistic historical documents is challenging, a deep model is introduced to transform initially generated synthetic data into refined data that appears realistic. The results show that pretraining on this realistic data reduces binarization error by 13% compared to no pretraining and reduces error by 6% compared to pretraining on unrealistic synthetic data.

Part II presents contributions for two segmentation tasks. In Chapter 5, we present a model that formulates the task of segmenting a document page from its surroundings as a pixel-labeling semantic segmentation task with domain-specific post-processing. On test data from the same collections as the training data, the model approaches the performance level of human agreement. On out-of-domain data, it still achieves good results, showing the generalizability of the model. Chapter 6 deals with the same problem, but with a generic keypoint regression model that has applications outside of document images. This model is able to outperform the model presented in Chapter 5 by directly regressing the image coordinates of the document corners from detection heatmaps. Segmentation of table images (and PDFs) into cells, rows, and columns is the subject of Chapter 7, where 2 deep models are introduced. These models use pooling regions appropriate for large table images and achieve state-of-the-art results on both the public benchmark dataset as well as a much larger and more complex private dataset.

In Part III, Chapter 8 presents an empirical analysis of CNNs used for document image page classification. While natural image classification can be done over lower resolution images, we find that higher resolutions lead to better document image classification. Using this and other insights, we were able to obtain state-of-the-art performance on a public dataset of 400K images, exceeding the performance of previous models pretrained on natural image datasets. In Chapter 9, we show that the same model architecture can be applied to both optical font recognition as well as medieval scribal script classification and obtain state-of-the-art performance. We identify some of the short comings of the basic classifier on the scribal script task and introduce a novel form of data augmentation to address some of the issues. Finally, we introduce a transfer learning method for handwriting recognition (HWR) in Chapter 10. While HWR models are very successful when trained on large amounts of labeled data, not all languages have sufficient amounts of labeled data to train such models. Chapter 10 shows that using only unlabeled text images and a language model in the target language is sufficent to enable transfer learning of a model pretrained on a source language that is similar to the target language.

References

[1] PDF converter to excel and word with OCR — ABBYY FineReader 14. https://www.abbyy.com/en-us/finereader/convert/. Accessed: 2018-12-17.

[2] How many languages in the world are unwritten? https://www.ethnologue.com/enterprise-faq/how-many-languages-world-are-unwritten-0. Accessed: 2018-12-17.

[3] Edmund Fournier d'Albe. On a type-reading optophone. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 90 (619):373–375, The Royal Society, 1914.

[4] Herbert F. Schantz. *The history of OCR, optical character recognition.* Recognition Technologies Users Association Manchester, VT, 1982.

Part I

Binarization of Historical Documents

This part presents 3 chapters on the topic of historical document binarization, where images of noisy text are converted to an image with only 2 allowed intensity values: black and white. Such pre-processing can make the later tasks of segmentation and recognition easier. Chapter 2 presents an extensive literature review of the binarization field including binarization methods, pre-processing, post-processing, parameter tuning, datasets, construction of ground truth, and evaluation methods. Chapter 3 presents a multi-scale Fully Convolutional Network (FCN) model for binarization that is trained using a proposed loss function to encourage errors to happen less frequently on important image pixels. One of the limitations of using deep learning for binarization is the amount of training data available. Chapter 4 shows that realistic synthetic data can be generated using deep networks and then used to pretrain models, such as the FCN from Chapter 3, to improve binarization performance.

Chapter 2

A Literature Review of Binarization of Historical Documents

Abstract

This review provides a comprehensive view of the field of historical document image binarization with a focus on the contributions made in the last decade. After the introduction of a standard benchmark dataset with the 2009 Document Image Binarization Contest (DIBCO), research in the field accelerated. Besides the standard methods for image thresholding, preprocessing, and post processing, we review the literature on methods such as statistical models, pixel classification with learning algorithms, and parameter tuning. In addition to reviewing binarization algorithms, we discuss the available public datasets and evaluation metrics, including those that require pixel-level ground truth and those that do not. We conclude with recommendations for future work.

Submitted to *Pattern Recognition.*

Chris Tensmeyer and Tony Martinez. "Historical Document Image Binarization - A Review."

2.1 Introduction

Binarization is the process converting a multi-tone image into a bi-tonal image. In the case of document images, it is typical to map foreground text pixels to black and the rest of the image (background) to white. In many applications, binarization is a critical preprocessing step and helps facilitate other document processing tasks such as layout analysis and character recognition. In such pipelines, the quality of the binarization can greatly affect system performance, as errors made in the binarization step can propagate to downstream tasks. As a standalone application, binarization can serve as a noise removal process to increase document readability. The file size of binary images is often orders of magnitudes smaller than the original gray or color images, which makes them cheaper to store on disk. Additionally, with the rise of digital archives, file size can become a concern as large numbers of images are viewed over the Internet. If a person can still recognize the text in the binary images, then this compression can be obtained with virtually no loss in semantic image content.

In the last decade, a tremendous amount of progress has been made in the field of historical document binarization. In 2009, the first Document Image Binarization Contest (DIBCO) introduced the first dataset of real degraded images that have ground truth annotations at the pixel level [36]. This enabled a standardized evaluation procedure that allowed for direct comparison between algorithms. This spurred research in the field and the creation of more datasets in new application domains.

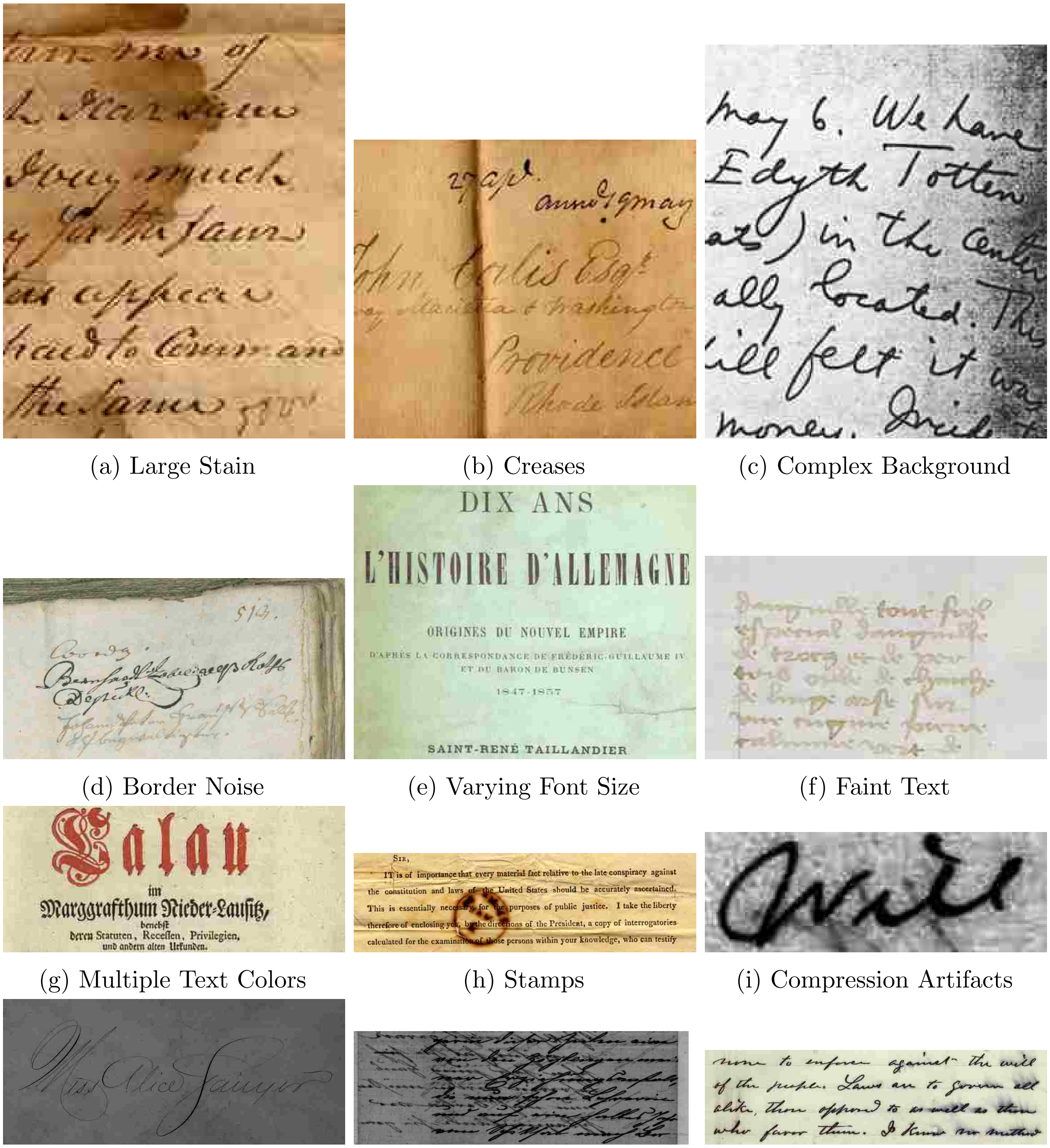
The purpose of this review is to highlight the recent advances of the last decade in all facets of historical document binarization. We discuss not only binarization methods, but also topics such as pre-processing, post-processing, algorithm efficiency, datasets, evaluation, and definitions of ground truth.

2.1.1 Challenges

Historical document images are, in general, more difficult to binarize than modern scanned documents. This is in part due to their degraded state and partly because many historical documents are digitized with cameras, which do not have controlled illumination conditions like scanners. Some camera produced images have uneven illumination due to bad lighting or because the page is not flat (e.g. curved edges near book bindings).

Figure 2.1 illustrates several degradation types that commonly occur in historical documents. Stains serve to decrease the contrast between foreground and background inside the stained region, and strong edges around stains can be difficult to distinguish from real foreground/background boundaries. Creases cause similar difficulties as stains. Complex backgrounds arising from paper textures or uneven illumination cause difficulties for statistical features because the statistics vary from one image region to another. Border noise was a major challenge in HDIBCO 2018 [120] because dark surroundings have similar intensity as foreground ink and book edges provide high frequency edges. Many local thresholding approaches use a fixed window size which is not optimal for handling various font sizes and stroke widths. Faint text is challenging due to the low contrast, and multiple text colors can lead to low intensity for some colors when converting to grayscale images. Stamps are similar to stains, but can be darker. Document images are high resolution and image compression (e.g. JPEG) is often used to meet limited storage requirements, but compression artifacts introduce high frequency content into background regions. Thin strokes (e.g. single pixel wide) can be discarded during image smoothing or noise removal processes. Bleed-through noise is perhaps the most challenging degradation because the noise is shaped like text, is superimposed over the foreground text, and may co-occur with faint non-bleed-through text. Finally, smeared ink can make it difficult to localize foreground/background boundaries.

Due to the large variety of historical documents, it is commonly accepted that no single algorithm can perform best on every image [70]. For example, in [54] it was noted that the state-of-the-art methods for paper documents do not produce good results for Palm Leaf Manuscripts (PLMs). Thus many binarization methods are designed to address some particular degradation type or domain.



(a) Large Stain (b) Creases (c) Complex Background

(d) Border Noise (e) Varying Font Size (f) Faint Text

(g) Multiple Text Colors (h) Stamps (i) Compression Artifacts

(j) Thin Strokes (k) Bleed Through Ink (l) Smeared Ink

Figure 2.1: Challenging aspects of historical document binarization illustrated with DIBCO images.

2.1.2 Previous Reviews

Previous reviews on historical document binarization have taken a much narrower scope than the current article. There are some short review articles that briefly describe about a dozen methods [11, 90, 91]. Other articles do not presents themselves as surveys, but do describe and then empirically evaluate a large number of existing algorithms [52, 137, 147].

A seminal review by Sezgin and Sankur [139] was published in 2004 which described 40 algorithms and provided detailed performance analysis on synthetic data. However, the binarization field has progressed much in the last 15 years and a new review is needed. Ismail et al. [47] recently reviewed 29 thresholding algorithms and organized them into a taxonomy based on what features are used to determine thresholds. While this review provides insight into thresholding methods, it does not explicitly address the many other types of methods and facets of the binarization field. In this review we give a comprehensive view of the field including non-thresholding methods such as Conditional Random Fields (CRF) and deep learning models. Additionally we cover facets besides binarization methods such as algorithm efficiency, parameter tuning, datasets, evaluation, construction of binarization Ground Truth (GT), and alternatives to GT evaluation.

2.1.3 Paper Organization

In contrast to prior reviews that provide individual paper summaries, the this paper is organized by topic. This choice is motivated by the modular nature of many binarization pipelines that combine many operations to achieve top performance (e.g. [103]). An individual summarization of each method would be insufficient to disentagle which specific parameters and operations (e.g. local threshold computation, noise removal, edge detection) are performance crucial and which may be interchanged without degrading binarization quality. Therefore, within each topic, we compare and contrast analogous operations and their relative merits. Fortunately, the community has recognized the value of comparing individual components and we draw on the analysis in the literature. We hope to highlight the contributions of individual operations that in isolation do not yield state-of-the-art performance, but may do so when incorporated into a full pipeline.

The remainder of the paper is organized in the following sections. Section 2.2 covers common preprocessing techniques for noise removal, background estimation, and grayscale conversion. State-of-the-art binarization methods are discussed in Section 2.3. Post-processing techniques, including morphology and how to combine multiple binary outputs are found in Section 2.4. Efforts for efficient binarization methods are discussed in Section 2.5, and datasets with ground truth (GT) are presented in Section 2.6. In Section 2.7, we review standard pixel-GT based evaluation, including GT construction and common metrics. Concerns about pixel-GT and alternative evaluation methods are discussed in Section 2.8. We offer concluding remarks in Section 2.9.

2.2 Preprocessing

The primary reason binarization is difficult is the presence of noise, degradations, and low contrast between foreground and background. Preprocessing techniques do not directly produce a binary image, but are designed to deal with these challenges in order to make subsequent binarization easier. Though there are many preprocessing techniques, they can be roughly categorized into noise removal, background estimation, and color to grayscale conversion.

2.2.1 Noise Removal

The goal of noise removal is to smooth background and foreground textures, while preserving the contrast between the two regions. This makes subsequent binarization easier as local image patches are more homogeneous.

Wiener filtering [160] is among the most common noise removal techniques used in binarization [35, 57, 79, 81, 136]. For known additive Gaussian noise, it produces an optimally filtered image according to the Mean Squared Error (MSE) criteria. Gatos et al.

[35] implemented Wiener filtering in the spatial domain as

$$I'(i,j) = \mu(i,j) + \frac{(\sigma^2(i,j) - \nu^2(i,j))(I(i,j) - \mu(i,j))}{\sigma^2(i,j)} \tag{2.1}$$

where $\nu^2(i,j)$ is a local estimate of the variance of additive Gaussian noise. In [35], $\nu^2(i,j) = \sum_{i'}\sum_{j'}\sigma(i',j')$, though as noted in [145], fixing ν^2 as the median $\sigma^2(i,j)$ computed over the whole image leads to better binarization performance.

Simple alternatives to Wiener filtering include low-pass Gaussian filtering [26, 106] and Bilateral filtering [3, 99]. Roe and Mello [129] proposed an interesting variation on a Bilateral filter, where each pixel is set to the median of its neighbors that have sufficient color-similarity. Other low pass filtering is done by truncating high frequency information through image transforms such as the Contourlet transform [165] and a Gabor-Wavelet decomposition [93].

Total Variation (TV) regularization [69, 145] finds a smooth image that resembles the original image. It is formulated as

$$\begin{aligned} I_f = \operatorname*{argmin}_{I'} &\sum_i \sum_j (I'(i,j) - I(i,j))^2 + \\ &\beta \sum_i \sum_j |I'(i,j) - I'(i-1,j)| + |I'(i,j) - I'(i,j-1)| \end{aligned} \tag{2.2}$$

where β controls the strength of the edge L_1 magnitude penalty. Setting β too high, however will cause undesirable smoothing of the edges that separate foreground and background. Other regularization terms, such as gradient L_2 and L_0 [48] can be used, but can be more difficult to optimize.

Non-local Means [22, 69, 145] smooths an image by performing a weighted average over neighborhoods, where the averaging weights are determined by patch similarity.

$$I_f(i,j) = \sum_{i'j' \in N(i,j)} w(i,j,i',j') I(i',j') \tag{2.3}$$

where $N(i,j)$ returns the (perhaps global) neighborhood of (i,j) and $w(i,j,i'j')$ is a weight based on the similarity between the image patches centered at (i,j) and (i',j'). The weights for any given pixel must sum to 1, i.e. $\forall i,j \sum_{i'j'} w(i,j,i',j') = 1$. It is known that averaging many noisy corruptions of an image (e.g. consecutive video frames) with zero-centered i.d.d. noise converges to the uncorrupted image. Non-local means attempts to average out the noise over similar local patches rather than over several images.

Kligler et al. [60] proposed a unique noise removal technique where the intensity image is converted to a point cloud $\{(x,y,I(x,y)\}$ that is embedded on a sphere. The transformed image values are based on the visibility of each point on the sphere from the center of the sphere. This preprocessing combined with Howe's method [46] achieved state-of-the-art performance on DIBCO data.

2.2.2 Background Estimation

Binarization of images with non-uniform background is often addressed by first estimating a grayscale background image and then using that estimate to preprocess the original image. One common way to compensate for the background is simple subtraction [21, 35, 64, 77, 99, 145, 156]

$$I'(i,j) = I(i,j) - B(i,j) \tag{2.4}$$

where B is the estimated background image. While subtraction is effective, it tends to leave low contrast in darker regions of the background due to small differences between foreground and background. This motivates dividing by the background image [49, 58, 63, 74, 103]

$$I'(i,j) = \frac{C}{B(i,j)} I(i,j) \tag{2.5}$$

where C is a constant used to scale the contrast. When $B(i,j)$ is dark (low intensity), $\frac{C}{B(i,j)}$ will be large and will serve to emphasize the foreground text in this region. In some cases, histogram stretching is performed on the normalized image to increase contrast [77, 129].

There are a variety of ways to estimate background images. A simple method is to use a median filter with a sufficiently large window size. Nina et al. [99] use a fixed 21x21 window size, while Mitianoudis and Papamarkos [84] progressively grow the window size at each pixel until the local standard deviation stabilizes. Morphological grayscale closing [21, 64, 77, 78] and heuristic estimation [145] have also been used.

Another popular technique is to perform a high-recall initial binarization and then inpaint the foreground regions based on neighboring background intensities [35, 57, 103, 106]. Using simple binarization techniques (e.g. Niblack [97], Sauvola [135]) tuned for high-recall is common choice for rough foreground estimation.

The winner of DIBCO 2009 proposed iterative polynomial smoothing [74], but this produces a highly smoothed background estimate. Perret et al. [113] used hypercomponent trees for background estimation, which combined with simple global thresholding achieved similar performance as [74]. Vo et al. [157] proposed robust regression for background estimation with some success.

2.2.3 Grayscale Conversion

The majority of binarization algorithms operate over grayscale images, so RGB color images are generally converted to grayscale before binarization. While most works employ simple conversion methods, such as the simple luminosity formula $g = 0.21r + 0.72g + 0.07b$, this can lead to decreased contrast for colored foreground ink. This problem is compounded when there are multiple foreground ink colors. For example, Hedjam et al. [44] report improvements of about 5-15% across 11 binarization methods when using their proposed grayscale conversion method on images with multiple and/or non-standard foreground ink colors.

One choice for conversion is to perform Principle Component Analysis (PCA) over the set of RGB pixels and project each pixel onto the first principle component [84, 85, 108]. Because the majority of pixels in any document image are background, the first principle component is close to the background color. Then after projection, background pixels will have high intensity (lighter) and foreground pixels will have low intensity (darker).

Hedjam et al. [44] proposed learning a fixed linear transform from a dataset of color images that have pixel GT labeling for text/background. The pixel features are its values in the RGB, YIQ, YDbDr, YCbCr color spaces. Then an optimization procedure learns the optimal linear mapping to grayscale such that average text and background intensities are separated. While their results are impressive, once learned, the mapping is fixed and only reflects the text and background colors in the training dataset.

Bouillon et al. [14] proposed a grayification method that is image-adaptive. First, pixel values are clustered in the YPQ color space. Then the intensity of each pixel is taken as the Mahalanobis distance between the pixel and the center of the largest (presumably background) cluster. This approach is similar to the PCA approach as the Mahalanobis distance depends on the principle components, but this method uses all of the principle components instead of just the first one.

We note that there are a number of (non-trivial) grayscale conversion methods proposed in the literature for photographic conversion that have been relatively unexplored in the context of document image binarization.

2.3 Binarization Methods

This section surveys many of the binarization methods proposed in the last decade and constitutes the bulk of this review. While most previous reviews proceed by summarizing methods in isolation, we have attempted to organize topically according to common themes.

2.3.1 Classic Thresholding Algorithms

There are a number of classical thresholding algorithms for binarization that are simple to implement and have become standard tools that many later algorithms use as sub-routines.

Otsu

The global thresholding algorithm proposed in by Otsu [105] remains popular, likely because it effectively handles images with uniform background and has no parameters to tune. The Otsu threshold, T_{Otsu}, is derived from the histogram of grayscale image intensity values, h, which typically has $H = 256$ bins for 8-bit images. Any chosen threshold $0 \leq T \leq L$ partitions the histogram into two clusters. The number of pixels, mean intensity, and variance of both clusters are respectively given by

$$w_0(T) = \sum_{i=0}^{T-1} h(i) \qquad w_1(T) = \sum_{i=T}^{L-1} h(i) \tag{2.6}$$

$$\mu_0(T) = \frac{1}{w_0} \sum_{i=0}^{T-1} ih(i) \qquad \mu_1(T) = \frac{1}{w_1} \sum_{i=T}^{L-1} ih(i) \tag{2.7}$$

$$\sigma_0^2(T) = \frac{1}{w_0} \sum_{i=0}^{T-1} h(i)(i - \mu_0(T))^2 \qquad \sigma_1^2(T) = \frac{1}{w_1} \sum_{i=T}^{L-1} h(i)(i - \mu_1(T))^2 \tag{2.8}$$

T_{Otsu} is then defined as the threshold that minimizes the within-cluster variance:

$$T_{Otsu} = \operatorname*{argmin}_T w_0(T)\sigma_0^2(T) + w_1(T)\sigma_1^2(T) \tag{2.9}$$

or equivalently maximizes the between-cluster variance, which reduces to

$$T_{Otsu} = \operatorname*{argmax}_T w_0(T)w_1(T)(\mu_1(T) - \mu_0(T))^2 \tag{2.10}$$

Finding T_{Otsu} is done by trying all values of T and seeing which one minimizes Eq. 2.9 or maximizes Eq. 2.10. Afterward, binarization is performed globally:

$$B(i,j) = \left\{ \begin{array}{ll} 0 & I(i,j) < T_{Otsu} \\ 255 & I(i,j) \geq T_{Otsu} \end{array} \right\} \tag{2.11}$$

Niblack

One disadvantage of Otsu or any other global thresholding method is that the background may be non-uniform. Some background pixels being darker than some foreground pixels, meaning that no global threshold exists to separate them. Niblack proposed a simple local adaptive threshold, where a threshold is determined for each pixel based on statistics computed from a local window centered on the pixel of interest [97]. Specifically, Niblack thresholding uses the local mean and local standard deviation:

$$\mu(i,j) = \frac{1}{w^2} \sum_{i'=i-w}^{i+w} \sum_{j'=j-w}^{j+w} I(i',j') \tag{2.12}$$

$$\sigma(i,j) = \sqrt{\frac{\sum_{i'=i-w}^{i+w} \sum_{j'=j-w}^{j+w} (I(i',j') - \mu(i,j))^2}{w^2}} \tag{2.13}$$

where w is called the window size and controls how much context is used to compute these statistics. The per-pixel Niblack threshold is then

$$T_N(i,j) = \mu(i,j) + k\sigma(i,j) \tag{2.14}$$

where k is a user-set parameter that controls the trade-off between foreground detection precision and recall. The recommended parameter setting is $k = -0.2$, though the optimal k depends on the image and chosen window size.

Binarization is then accomplished with

$$B(i,j) = \left\{ \begin{array}{ll} 0 & I(i,j) < T_N(i,j) \\ 255 & I(i,j) \geq T_N(i,j) \end{array} \right\} \tag{2.15}$$

The biggest issue with Niblack thresholding is when the window covers only background pixels, causing the darkest background pixels to be set to foreground.

Sauvola

Sauvola and Pietikäinen [135] proposed a variant of Niblack to solve the problem with background-only windows.

$$T_S(i,j) = \mu(i,j)\Big[1 + k\big(\frac{\sigma(i,j)}{R} - 1\big)\Big] \tag{2.16}$$

where $\mu(i,j)$ and $\sigma(i,j)$ are computed as in Niblack (Eqs. 2.12,2.13), $k = 0.5$ is the recommended value for the user-set parameter, and R is a constant set to the maximum possible standard deviation, i.e. $R = 128$ for 256 gray levels.

While Niblack takes $\mu(i,j)$ and adjusts downward based only on the $\sigma(i,j)$, Sauvola adjusts downward based on $\mu(i,j)\sigma(i,j)$. In windows of only background $\mu(i,j)$ is relatively large, so $T_S < T_N$, which means fewer of these background pixels are set to foreground.

Wolf

Wolf binarization [161] is an extension of Sauvola, where the local statistics are normalized based on global statistics:

$$T_W(i,j) = \mu(i,j) - k(1 - \frac{\sigma(i,j)}{S})(\mu(i,j) - M) \tag{2.17}$$

where $S = \max_{ij} \sigma(i,j)$, $M = \min_{ij} \mu(i,j)$. This allows for better handling of images with limited contrast and limited range of grayscale intensity.

2.3.2 Image Processing

The majority of binarization techniques employ one or more image processing techniques. Here, we have roughly categorized these techniques into thresholding, edge detection, image transforms and region based methods, but we recognize that much of the preprocessing (Section 2.2) and postprocessing (Section 2.4) techniques also involve image processing.

Thresholding

Inspired by classic algorithms, many approaches are based on local thresholding, sometimes combined with preprocessing. One competitive example of this is by Gatos et al. [35], where after background subtraction (Section 2.2.2), the local threshold is based on an estimate of the average distance between foreground and background pixels. Then to compensate for low contrast regions, the threshold is adapted using a logistic sigmoid function to produce lower thresholds when the estimated background region is dark.

NICK thresholding [59] (named for the authors initials) perturbs $T_{Niblack}$ based on the global image mean, μ, to lower the number of false positive foreground components. From Eq. 2.14, we have $T_{Niblack}(i,j) = \mu(i,j) - k\sqrt{B}$, where $B = \sigma^2(i,j)$. The NICK threshold has a similar form with $T_{NICK} = \mu(i,j) - k\sqrt{A}$, where $A \approx B + \mu^2$ for large window sizes. Bataineh et al. [9] proposed a local threshold computed from both the local and global means and standard deviations that was shown to outperform NICK on DIBCO 2009, though parameters were not necessarily tuned for NICK. Adaptively setting the k value in NICK thresholding based on the global standard deviation was shown to improve performance over a fixed k [133].

While most local thresholding methods use all pixels in the local window, [49] as well as the winners of DIBCO 2009 [75] and 2013 [151] determine local thresholds based on the intensity of detected edge pixels. Additionally, in [75], pixels classified as foreground must be near N_{min} number of edge pixels and have an intensity below the average nearby edge pixel. Estimating the stroke width (see Section 2.3.5) is often used to determine the window size for edge-based local thresholding.

AdOtsu [87] is a local version of Otsu (Section 2.3.1) where the threshold is computed from a local window rather than the global histogram. In addition, there is a switching behavior to use an alternative constant threshold when regions are similar to the estimate background image.

Edge Detection

Several approaches perform edge detection to find the boundary between foreground and background. A popular method is Canny edge detection [19], which finds single pixel wide edges by performing non-maximal suppression and hysteresis thresholding over the gradient magnitude image computed from the input image after Gaussian smoothing. The hysteresis thresholding process first finds strong edges using a high user-set threshold, τ_1, and then expands each strong edge to include edge pixels with gradient magnitude above τ_2, where $\tau_1 > \tau_2$. The winner of DIBCO 2013 [151] filtered edges detected by Canny using a novel measure of local contrast:

$$C(i,j) = \alpha \frac{M(i,j) - m(i,j)}{M(i,j) + m(i,j) + \epsilon} + (1-\alpha)(M(i,j) - m(i,j)) \tag{2.18}$$

where $M(i,j)$ is the image local maximum, $m(i,j)$ is the image local minimum, α increases with the global image standard deviation, and ϵ is a small constant to prevent division by 0 [151]. Without manually tuning parameters, it is difficult to eliminate false edges with Canny, and while $C(i,j)$ does a good job of eliminating false positives, it only roughly localizes the edges, even if a small window size is used to compute M and m. After this process, spurious horizontal edges may still remain in Palm Leaf Manuscripts (PLMs) due to the background texture of palm leaves, but many of these edges can be removed by examining the aspect ratio of connected edge components [112].

Binarizing based on pairs of Structural Symmetrical Pixels (SSPs) leverages the idea that edge pixels on opposite sides of the stroke should have large gradient magnitudes and opposite orientations [49]. While spurious edges may be detected due to background variations, the noise edges are less likely to be symmetrical and thus are not SSPs. An adaptive water flow model is proposed in [154], where water is poured on detected edges and allowed to flood low intensity basins at a rate and amount controlled by edge strength and stroke edge. After filtering spurious regions, all pixels covered by water are designated to be foreground.

One issue with using edge detection for binarization is that the detected edges do not completely surround the foreground regions. In contrast, active contour methods constrain the detected edges to be closed contours that directly imply a segmentation. Hadjadj et al. [39] uses Eq. 2.18 (with $\alpha = 1$) to initialize closed contours and then optimizes those contours according to forces that attempt to align the contours to maximize local intensity differences and minimize the contour length and area.

Image Transforms

Some methods have experimented with extracting local features based on image transforms such as Gabor filter banks, Fourier transform, and the Contourlet transform.

In [95], thresholding the response of log-Gabor filters is used to identify the pixels that correspond to maximal local phase components in the Fourier frequency domain. Sehad et al. [136] finds the dominate document slant angle using the Fourier transform and then constructs a weighted Gabor filter bank based on this angle to smooth the image. Then for local thresholding, they compute the local standard deviation from the Gabor-filtered image and compute the local mean from the input image.

The Contourlet transform is composed of a Laplacian pyramid decomposition to obtain multi-scale analysis and a directional filter bank is applied at each scale [165]. The resulting Contourlet coefficients give a more concentrated representation of the image edges, which are thresholded using Niblack. Then the inverse Contourlet transform is applied to produce a smoothed image that is fed to a local thresholding method.

2.3.3 Statistical Models

While the image processing methods presented in Section 2.3.2 can perform well, they aren't grounded in a formal framework. In this section, we review statistical methods based on

Mixture of Gaussians (MoG) and Conditional Random Fields (CRF), in addition to other methods based in probability theory.

Mixture Of Gaussians

A MoG model is a parametric probability distribution whose probability distribution function (pdf) is a weighted sum of Gaussian distributions,

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \, \Sigma_k^2) \tag{2.19}$$

where $\mathcal{N}$ is the pdf of a Gaussian distribution, K is the number of mixture components, and $\sum_{k=1}^{K} \pi_k = 1$. The MoG can be fit to a set of unlabeled data points (e.g. pixel features) using Expectation Maximization (EM). Afterward, each data point can be assigned to its most probable cluster, and the clusters can be assigned to either the foreground or background class to induce a binarization over the entire image.

How to apply the MoG in binarization varies among the published literature. For example, foreground and background have been modeled each with a single component using intensity or color as the pixel features [43, 67]. Both these methods extend the basic Gaussian mixture component to have a spatially varying μ_k, and [43] uses a spatially varying Σ_k. In this manner, a single component can model spatially varying changes in the statistics of foreground or background pixels. The FAIR algorithm [67] combines this MoG approach with some pre-processing and post-processing to obtain state-of-the-art results.

In contrast, Mishra et al. [83] uses 5 components for foreground and background and uses both stroke and color features. Though [83] primarily targeted scene text segmentation, where foreground text may be multiple colors, it performs competitively with the state-of-art on DIBCO datasets. Mitianoudis and Papamarkos [85] perform MoG clustering over pairs of intensity values that co-occur in local windows combined with a local contrast feature.

Though a MoG can converge to any arbitrary distribution (given enough components), an empirical study over handwritten images found that foreground and background patches far from character boundaries have intensities that are approximately Gaussian distributed [127]. However, including the (transition) pixels near boundaries creates skew in the foreground and background distributions. Due to this asymmetry, a mixture of log-normal distributions provides a better fit and increases binarization performance compared to a MoG [127].

Conditional Random Fields

CRFs provide an excellent framework for including spatial dependencies among pixels into the binarization process. A CRF is governed by an energy (cost) function that scores how well a given binarization agrees with the input image. For tractability, this energy function is decomposed into local energy functions over sets of pixels. This decomposition could be over sets of any size, though it is typically defined over single pairs and pairs of neighboring pixels:

$$E(B, I) = \sum_{i=1}^{N} E_i(B_i, I) + \sum_{i=1}^{N} \sum_{j=1}^{i} E_{ij}(B_i, B_j, I) \tag{2.20}$$

where I is the input image with N pixels, B is a discrete segmentation of I, and E_i, E_{ij} are respectively the decomposed unary and pairwise energy functions, which optionally can vary based on i or i, j. Though general CRFs can have any number of segmentation classes, for binarization it is typical to have 2 classes. Typically, E_1 biases individual pixels to be either foreground or background based on pixel intensity and E_2 yields low values when adjacent, similar pixels are assigned to the same class.

Given that Eq. 2.20 scores the goodness of a given binarization, we formulate the binarization inference task as energy minimization

$$\hat{B} = \operatorname*{argmin}_{B} E(B, I) \tag{2.21}$$

An exact solution to Eq. 2.21 can be found for binary segmentation tasks if E_{ij} satisfies the following sub-modularity criteria [61]

$$\forall i, j \; E_{ij}(0,0,I) + E_{ij}(1,1,I) \leq E_{ij}(0,1,I) + E_{ij}(1,0,I) \tag{2.22}$$

If the above is true, then E is can be transformed to a graph with non-negative edge weights, and a minimum cut over this graph corresponds to $\hat{B}$ [61]. The minimum graph cut problem can be reformulated as the dual max-flow problem, for which there are many algorithms and off-the-shelf solvers [15].

The popular Howe method [46] uses a CRF, where E_i is set to be the image Laplacian. The pairwise energy, E_{ij}, is 0 in uniform regions and discontinuities that occur at edges. Specifically, E_{ij} gives a uniform penalty when $B_i \neq B_j$ unless pixels i and j are on opposite sides of detected Canny edges [19]. The edges used to define E_{ij} are improved in [6] using an initial binarization obtained by clustering pixels. Howe's method (and variants to introduce pre- and post-processing) placed first in DIBCO competitions for 2012, 2014, 2016, and 2018 [102, 116, 118, 120].

In [111], a CRF is used to improve an initial binarization, where E_{ij} is determined by the distances between foreground pixels, background pixels, and the foreground skeleton of the initial binarization. Similarly, [150] performs an initial ternary classification into foreground, background, and uncertain before using a CRF to reclassify the uncertain pixels based on detected edges and similarity to nearby foreground and background pixels.

The previously described CRFs use hand-crafted energy functions and do not learn parameters. In contrast, Ahmadi et al. [2] defines E_i and E_{ij} as a linear combination of feature functions based on window statistics, Histogram of Oriented Gradients (HOG) [27], Local Binary Patterns (LBP) [104], and Canny edges [19]. The linear combination weights (feature importance) are then learned from training data and fixed for novel test images.

Other Statistical Approaches

The idea of a transition pixel was introduced in [122] and later expanded upon [123, 126]. These works define a statistical criteria for modeling pixels near foreground-background transitions that is used for image thresholding.

A level-set method for binarization is presented in [128], where the contours separating foreground and background are evolved according to forces that direct the contours to be smooth, have minimum area, and adhere to strong edges. One of the proposed forces is based on modeling foreground and background intensities using locally-linear models and the contours are adjusted to increase the likelihood of the segmentation under this model.

2.3.4 Pixel Classification

In the last few years, machine learning methods, particularly deep learning models, have become the state-of-the-art in historical document binarization. In 2017, the top 6 submissions to DIBCO all used a deep model [119].

The majority of learning methods adopt a framework of directly classifying pixels, while others use machine learning to predict other quantities such as optimal parameter settings or methods. The latter approach is covered in Section 2.3.5 and the pixel classification approaches are covered here. First we shall review shallow models and then deep models.

Shallow Models

Hamza et al. [40] used a Self-Organizing Map (SOM) to cluster pixels with similar intensity and a Multi-Layer Perceptron (MLP) to classify them as foreground or background based on intensity. An MLP was also used in [121] to classify individual pixels in Brazilian bank checks that have complex background due to check security features and watermarks. This MLP took as input the pixel intensity as well as the average intensity of 4 3x3 windows around the pixel. In contrast, the MLP in [53] used the raw intensities in a 3x3 window and the global mean and standard deviation to classify the central pixel. However, such small context windows restrict the model to learn very simple features that do not generalize across a wide variety of documents. Pastor-Pellicer et al. [110] used 90 inputs composed of the 81 raw intensity values of a 9x9 window plus 9 intensity values of the estimated background for a 3x3 window. In addition, the authors proposed a new loss function which is a continuous adaptation of the F-measure evaluation metric and showed this resulted in improved MLP training.

Ensembles of shallow models have also proven effective in classifying pixels. In [51], an ensemble of 8 Support Vector Machines (SVMs) are trained to classify pixels in different positions in a 3x3 window, leading to an error reduction of 25% compared to a single model. Wu et al. [162] used a variant on a Random Forest ensemble combined with a large, rich feature set to achieve competitive performance. Their features include thresholds determined by classic algorithms, statistics over multiple scales, and histograms for several (non-centered) regions. Another key element was constructing a training set of difficult pixels based on the disagreement regions of other algorithms.

Deep Neural Networks

While early uses of deep Convolutional Neural Networks (CNN) involved small images for character recognition [65, 142], they became popular for large image classification after a CNN was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [132]. In [72], CNNs were transformed to Fully Convolutional Networks (FCNs) for semantic segmentation, which is defined as a task where each pixel is classified as one of K classes. Binarization is a semantic segmentation problem with $K = 2$ and methods based on FCNs have proven very effective on this task. However, the basic architecture of [72] only obtains 68% F-measure on DIBCO 13 and 14 [157] and has been extensively modified by the following approaches.

Pastor-Pellicer et al. [109] first explored a CNN for binarization, finding that a CNN with an appropriate architecture outperforms a non-convolutional deep MLP even if the MLP is augmented with additional histogram and median filter features.

Both Peng et al. [112] and Calvo-Zaragoza and Gallego [18] adopted an encoder-decoder architecture, where the encoder progressively downsamples the input image to pool information over increasingly larger input regions. The decoder then performs upsampling to produce a binarization at the original input resolution. In [112], the encoder-decoder is trained on synthetic data in steps, where additional layers at a smaller resolution are added during training. Then each scale makes a prediction, which is combined using a CRF. Calvo-Zaragoza and Gallego [18] explored 3 basic network blocks and found that residual blocks perform the best. They also find a large gap in performance between models trained on specific document domains (e.g. DIBCO, PLM, PHIBD) and a generalist model.

Tensmeyer and Martinez [153] and Vo et al. [157] proposed FCNs that learn features at multiple scales. While the single model proposed in [153] fuses features learned in parallel at multiple scales, in [157] there are 3 separate networks trained independently at different input image scales. The predictions of the 3 models are combined by taking the union of the individually predicted foreground pixels, which allows for both fine and coarse prediction.

The previously discussed FCNs do nothing to enforce spatial smoothness on the output predictions, which has led to some post processing steps. For example, the model of [153] was submitted to DIBCO 2017 [119] both with and without fully connected CRF post processing [62]. In contrast, PDNet [7] integrates individual pixel probability prediction with a smoothness term based on total variation. This improves performance by jointly learning (end-to-end) the pixel probabilities and output smoothness, rather than smoothing outputs in a post-hoc fashion.

Recurrent Neural Networks (RNNs) for binarization have also been proposed, though they are not as effective as FCN based approaches. In [1], 4 small 2D BLSTMs (Bidirectional Long-Short Term Memory) are used to model an arbitrarily large context for pixel classification.

The 4 models process the image in different directions (top-down, left-right; top-down, right-left; down-top, left-right; down-top, right-left) and fuse their features before making a prediction. Westphal et al. [159] instead use Grid LSTMs and report performance near that of [157]. In Grid LSTMs, each pixel has a set of memory cells for both vertical and horizontal directions.

Class Imbalance

One classic problem for machine learning algorithms is when the training and/or test data is heavy skewed to one class (e.g. background pixels). This is addressed in [110] by training to optimize the F-measure, which causes the model to balance the false positive and false negative errors it makes. In [153], the pseudo F-measure (see Section 2.7.2) loss is used for training, which heavily penalizes errors made near the foreground text. Similarly, [159] uses the per-pixel weights from pseudo F-measure to weight the per-pixel contribution for a cross-entropy loss.

Another method to address class imbalance is to subsample the background class. This is done in [162] by sampling a training set from disagreement regions of other algorithms. In [33], training patches are sampled according to the standard deviation of intensity, which oversamples foreground regions.

2.3.5 Parameter Tuning

Many binarization methods have tunable parameters such as window sizes or constants used in computing thresholds. The output of these algorithms is generally sensitive to these parameters. For example, the reported F-measure for Sauvola on DIBCO 2009 ranges from 69.54 [34] to 87.26 [37] in the literature, presumably due to different parameter settings used by authors. Even the choice of which binarization algorithm to use for a particular image can be considered a meta-parameter, as it has been repeatedly shown that no single algorithm performs best for all images [70]. Parameter tuning approaches come in 3 flavors: supervised, unsupervised, and interactive.

Supervised

In supervised parameter tuning, there are two primary approaches taken. In the first, a set of training images is used to determine a static set of parameters, which are then applied to each new image without modification. The second approach is more adaptive in that the training set is used to learn a model that predicts the optimal setting of parameters conditioned on the image. These approaches are similar to pixel classification (Section 2.3.4), but indirectly infer pixel classes by instead predicting parameters.

The static scenario can be posed as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} M(f(I_i; \theta), B_i) \tag{2.23}$$

where θ is the set of parameters being optimized, f is a particular binarization algorithm, $\{(I_i, B_i)\}$ is a training set (size N) of input images I_i with corresponding GT B_i, and M is a metric to be maximized such as F-measure. While a brute force search is viable for a limited number of parameters (as it only needs to be done once), more efficient search strategies have been employed in the literature. These include I/F Race [78], Bayesian optimization [155], and swarm optimization techniques [30, 41].

A general formulation of the conditional setting is

$$\hat{\psi} = \underset{\psi}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} M(f(I_i; g(I_i; \psi)), B_i) \tag{2.24}$$

where g is a learning model with parameters ψ that predicts $\theta_i = g(I_i; \psi)$. For a test image, I, the prediction is made by $f(I, g(I; \hat{\psi}))$. Cheriet et al. [24] predicted parameters for Sauvola and Lu [74] binarization methods, attaining results very close to those obtained using optimal parameter settings. Chattopadhyay et al. [20] segments an image into blocks and then uses a SVM classifier (trained to optimize OCR performance) to choose the best binarization algorithm to apply to each region based on region statistics. A similar approach was used in [164], but instead a global threshold is predicted for each block. Westphal et al. [158] replace the unsupervised parameter tuning of Howe [46] with a supervised approach and report improved performance.

An alternate formulation where g is selected based on a training and validation set is presented in [80], and [79] uses a handcrafted g to detect 4 types of image noise and provide appropriate parameters for the detected noise type.

Unsupervised

In unsupervised parameter tuning, parameters are tuned using only the information contained in the test image. A common parameter that is estimated in this way is the average stroke width [13, 71, 74, 83, 86, 154]. One way to estimate the stroke width is to perform an initial binarization (e.g. Sauvola) or edge detection with Canny. Then create a histogram for the distances between each pair of successive edge pixels on all horizontal scan lines. Then the stroke width can be estimated as the most frequent distance between the edge pixels [13, 86]. In [49], this is done in an iterative fashion as the binarization result is based on the stroke width and the stroke width is estimated from the binarization.

The window size is often based on the stroke width [103], but other ways are used to estimate it based on the idea that the window should be large enough to cover both foreground and background pixels. In [9], pixels are initially sorted into foreground, background, and low-confidence. The window size for thresholding low-confidence pixels is chosen to ensure that a sufficient number of foreground pixels are included in each window. The methods of [12, 26, 84] gradually increase the window size until reaching a stopping criteria based on window standard deviation.

Howe [46] tunes two critical parameters to minimize a criteria that measures binarization output instability. In [68], the weighted average of a globally and a locally computed threshold is determined by minimizing a cost function. Several statistical cost functions were evaluated in [124] and found to correlate well with OCR performance.

Interactive

Interactive parameter tuning requires the input of a user to improve binarization performance for a particular image. In [75, 76], the user is able to provide feedback to the algorithm by indicating approximate regions that were poorly binarized. This feedback can take the form of scribbles around the region [75], which after segmentation is binarized again with an adjusted threshold. The user feedback in [76] is class-based, where the user indicates small regions that are foreground, bleed-through, and background. This small feedback is propagated to other regions and a new binarization result is produced that can be further refined by the user.

Sokratis and Kavallieratou [146] proposed an interface that allows users to manually set parameters and provide global feedback for the resulting binarization. Based on the feedback, the interface suggests a modification to the parameters. The suggestions, however, are based on pre-determined rules, which limits its applicability in new algorithms.

2.4 Post-Processing

Once a binarization mask has been produced, it can often be improved by post-processing. This is mostly a heuristic process, involving morphological operations based on apriori knowledge of how binarization masks of text should look. Other techniques are based on comparing the resulting binary image to the input grayscale image or based on combining multiple binarized images.

2.4.1 Morphological Approaches

The main purpose of morphological post processing is to smooth the output binary image to remove noise. Generally, this involves removing small foreground components and smoothing the edges of other foreground components. To obtain the components, connected component analysis is performed using 4-connected or 8-connected neighborhoods.

Removing small foreground components is typically done by counting the number of pixels in the component and comparing it to a threshold. This threshold in the literature ranges from conservative 1-4 pixels [21, 108, 122, 151] to as large as 60 pixels [136]. Other methods set this threshold adaptively, such as 10% of the average foreground component size [82]. While using a larger threshold is likely to remove more spurious components, there are legitimate small foreground components such as the dot of an *i* or diacritic marks that may be removed by this process.

Smoothing the edges of foreground components can be done in several different ways. One simple way is to perform a morphological closing operation followed by a morphological opening operation [113, 136]. Lu et al. [74] explicitly remove single pixel concavities and convexities. Shrink and swell filters have also been used, where pixels are flipped if the number of opposite class pixels in a local window is above some threshold. In [35], the thresholds and window sizes were determined based on the average character height. To recover faint strokes, [22] constructs a bidirectional graph over extracted strokes and then applies a swell filter to try to connect strokes that are detected as broken.

Some methods that explicitly detect edge pixels ensure that pixels on opposite sides of the detected edge are different classes [21, 151].

2.4.2 Approaches Based on Grayscale Image

Besides morphology, other post-processing approaches have been proposed. For example, local thresholding based on both the binary and gray image has been used to fill in the interior of characters with thick strokes [10] and to connect broken strokes [8]. In [108], these regions are called *white islands* and are identified by performing a two sample z-test of the mean gray value of the white island and the mean gray value of the surrounding pixels. Other approaches for fixing misclassified background pixels involve comparing the intensity of each background pixel to the intensities in the surrounding window [57, 92].

2.4.3 Combining Multiple Binarizations

Combining the output of 2 or more binarization methods is a good way to improve overall binarization quality. One strategy is to compute two binary images where one has high precision of foreground components even if it does not capture the full extent of each component [4, 10, 38, 103, 108, 125]. The other image has high recall, but could have many false positive foreground components (e.g. the output of Niblack). These images can be combined by retaining only the foreground components of the high recall image that have sufficient overlap with one of the components in the high precision image. This serves to filter the false positives to greatly improve the precision of the high recall image.

An unsupervised Ensemble-of-Experts method was proposed in [88], which analyzes the outputs from a large number of binarization algorithms to form a weighted consensus graph. The idea is to find the correct cluster of experts that agree on a given test image and produce a final binarization using a majority vote of the selected experts. Though an ensemble of all submissions to HDIBCO 2012 was unable to outperform the best system in terms of F-measure, an ensemble of Howe's method [46] using different parameter settings was able to outperform the parameters chosen by the automatic tuning [46]. Similarly, a supervised ensemble of experts combined with a trained CRF was shown to outperform the best of the ensemble [42].

Su et al. [149] proposed a way to combine the output of two binarization methods using features extracted from the input gray image to classify the pixels that the two binary outputs do not agree on. To incorporate more than 2 methods, the combined output of 2 methods can be compared to a third output, and so on. On DIBCO 2009, they were able to combine two methods to achieve an average F-measure of 93.18, where the individual binarization methods had average F-measures of 91.06 and 91.24. An extension of this method replaces the handcrafted classification rules for disagreement pixels with sparse reconstruction classification [152]. For each uncertain pixel, a small set of similar patches centered on pixels that the two methods agreed on are used to determine the class of the uncertain pixel.

2.5 Binarization Efficiency

Some binarization algorithms are too slow to process very large collections (e.g. 50 million [159]) or to be used interactively [28, 75]. For example, the winning algorithm of the 2014 HDIBCO took 17 seconds to process an average sized image [102]. Such considerations have pushed some researchers to search for ways to speed up binarization algorithms on both CPU and GPU.

2.5.1 CPU

The primary ways that CPU based algorithms have been made more efficient is through grid-based and integral image techniques.

Grid-based Methods

In grid-based methods, certain quantities (e.g. local thresholds) are computed for only a subset of pixels on a regular grid (i.e. all pixels whose coordinates are divisible by $N \in \mathbb{Z}^+$). Under the assumption that these quantities smoothly vary across the image, the value for non-grid pixels can be computed through interpolation, which is typically much faster than computing the exact quantity. In [86], grid-based Sauvola is proposed by computing the local threshold for only the grid pixels and inferring the rest of the local thresholds through interpolation. The grid approach was shown to be 640x faster and yielded improved binarization quality compared to basic Sauvola due to the imposed smoothness of the local threshold. While this was shown for Sauvola, this technique can be applied to other local thresholding methods and other quantities (e.g. [43]).

Integral Images

The integral image, L, of image I contains cumulative sums of I at each pixel:

$$L(i,j) = \sum_{i'=1}^{i} \sum_{j'=1}^{j} I(i',j') \tag{2.25}$$

Once L has been computed (in $O(HW)$ time), the sum of any rectangular region defined by (y_1, x_1, y_2, x_2) of I can be computed in constant time.

$$\sum_{i'=y_1}^{y_2} \sum_{j'=x_1}^{x_2} I(i',j') = L(y_2,x_2) - L(y_1,x_2) - L(y_2,x_1) + L(y_1,x_1) \tag{2.26}$$

For local thresholding methods that make heavy use of computing statistics over local windows, this can provide significant speed up (5-20x), reducing computational complexity from $O(HWK^2)$ to $O(HW)$, where K is the chosen window size [140].

Quantities other than sums can be computed by first transforming I before computing L. For example, the integral image of $I^2(i,j)$ can by used to compute the local variance [122, 140]. Integral images can also be used for vector valued transformations of I such as grayscale histograms [98].

2.5.2 GPU

Graphical Processing Units (GPUs) are designed for efficient Single Instruction Multiple Data (SIMD) parallel computing. The most common use of GPUs in binarization is for deep learning models (Section 2.3.4), which have a high computational burden (e.g. 125 GFLOPs [72]) The CuDNN library [25] provides efficient GPU routines for both training and inference, and it has been integrated into a variety of learning frameworks. However, the recently proposed deep models have not given much consideration to speed, so while they may win contests [119], their use in workflows that require high throughput or low latency may be limited. Techniques to make networks smaller and faster without losing accuracy have been studied in the deep learning literature (e.g. [130]).

Hybrid approaches that utilize both CPU and GPU have been investigated. In [23], Sauvola is applied to the image several times on the GPU with multiple sets of parameters. Then SVM based voting and reconstruction is performed on the CPU.

Westphal et al. [158] performed an extensive study of a CPU/GPU implementation of Howe binarization [46], finding that the fastest configuration performs all steps on the GPU, achieving a 3.5x speed up compared to the reference implementation. They implemented a parallel algorithm to solve the graph-cut energy minimization step on the GPU because the original solver used in [46] is a serial algorithm.

2.6 Datasets

A number of datasets for binarization of degraded documents with GT pixel labeling have been proposed and used for evaluation in the literature.

The main dataset used in the literature comes from the DIBCO series of competitions [36, 102, 114–120]. Each year since 2009 (except 2015), a new dataset of 10-20 handwritten/machine printed images has been released, totaling 116 images in 2018. The images are chosen to have "representative degradations" and come from a variety of collections and libraries [115]. All images contain text from a Latin-based language. In evaluation, some authors report results on each dataset, separated by year as to be comparable to the competition entrants, while others report averages across several years.

The Persian Heritage Image Binarization Dataset (PHIBD) [5, 94] contains 15 handwritten images of Persian writing with various degradations and has attracted less attention than the DIBCO data. The CMATERdb 6 dataset of 5 color images was proposed in [89], but is no longer available for download. The Bickley Diary dataset [151] consists of 7 annotated page images of a degraded 1920s diary and has been used in a handful of later works [58, 95, 157].

The Irish Script on Screen (ISOS) bleed through dataset [131] contains 25 registered verso/recto image pairs that are annotated at the pixel level with 3 classes: foreground, background, bleed through. Though intended for the related problem of bleed through removal, such a dataset could be used to evaluate the performance of binarization algorithms on this specific degradation.

2.6.1 Palm Leaf

Palm Leaf Manuscripts (PLMs) are characteristically different from the previous collections of paper documents. State-of-the-art algorithms for DIBCO datasets do not perform well on PLMs. For example, [54] reports that Howe's method achieves 42.47% F-measure, while the classic Niblack achieves 60.48% on a dataset of PLMs.

The AMADI_LontarSet contains 100 pages of Balinese PLMs, split into 50 images for training and 50 images for testing [56]. Each image was annotated by two different annotators to produce 2 versions of GT. An additional 46 Khmer and 61 Sudanese PLMs were released as part of the 2018 ICFHR competition [17]. These images appear distinct from the Balinese PLMs.

2.7 Evaluation with Pixel Ground Truth

The primary way that binarization algorithms are currently evaluated is by comparison to reference ground truth (GT) binarizations. In Section 2.7.1, we explain common ways to construct reference GT images. In Section 2.7.2, we give common evaluation metrics. Criticism of this method of evaluation and alternatives are presented in Section 2.8.

2.7.1 GT Construction

The semi-automated method used to annotate the DIBCO competition images was presented in [100, 101] and is summarized here. First an initial binarization is produced using an automated method (e.g. [35]). This initial binarization is skeletonized so that components are a single pixel thick. The user then removes false alarm components and draws skeletons for missed components. Then Canny edge detection [19] is applied to the input image and the user corrects the edge map. Finally, the skeletonized image is iteratively dilated, constrained by the edge image, using a 3x3 cross-type structuring element. Each component is iteratively dilated until 50% of the enclosing edge pixels are covered.

To annotate the AMADI_LONTAR dataset [54, 56] of Palm Leaf Manuscripts (PLMs) for binarization, the DIBCO framework [100] was followed. Because existing binarization algorithms performed very poorly on PLMs, a new semi-local binarization scheme was proposed.

To create the Persian Heritage Image binarization Dataset (PHIBD), a simpler semi-automated method was used [94]. First, the user selects the type of image (handwriting, machine print, combined) and the types of degradations present (e.g. bleed through). Then a phase based binarization method is applied with parameters chosen according to the user input. Then the user uses the PixLabler [134] tool to manually correct the automated binarization at the pixel level.

Synthetic Data

Stathis et al. [147] and Lins et al. [70] have proposed using synthetic data, created by compositing text and background, to evaluate binarization algorithms. The advantage of using synthetic data is that a large quantity of images can be used, and the foreground annotations are more objective than human-annotated real data. However, the disadvantage is that the data does not always resemble real-world data.

In [147], foreground is extracted from PDFs and is composited on authentic noisy background by taking the element-wise maximum or by average blending. The latter technique leads to a more natural looking result, but results in a lighter background due to the PDF background being white. This method was also used to construct the test set for the ICFHR 2010 Quantitative Evaluation of Binarization Algorithms [107].

Lins et al. [70] create synthetic backgrounds by first clustering background patches from 3351 documents to find 100 distinct background textures. Then novel backgrounds can be created through random sampling from a particular texture or through image quilting techniques for texture generation. Then foreground text and artificial bleed through are composited on the generated background through traditional image compositing. They use a parameter, α, to control the strength of the bleed through and evaluate algorithms at different levels of bleed through degradation.

Though not specific to synthetic data for binarization, the DocCreator library [50] contains degradation models for introducing ink fading, adaptive blur, bleed through, and uneven illumination. Seuret et al. [138] proposed inserting stains and spots using gradient domain image editing.

2.7.2 Metrics

A wide variety of metrics have been used to evaluate binarizations using pixel GT.

Standard Binarization Metrics

F-measure (FM) is the harmonic mean of precision (P) and recall (R), which in turn are defined by the number of True Positives (TPs), False Positives (FPs), and False Negatives (FNs).

$$FM = \frac{2PR}{P+R} \qquad P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \tag{2.27}$$

$$\begin{aligned} TP = \sum_{ij} B(i,j) \wedge G(i,j) \quad FP &= \sum_{ij} B(i,j) \wedge \neg G(i,j) \\ FN &= \sum_{ij} \neg B(i,j) \wedge G(i,j) \end{aligned} \tag{2.28}$$

where B is the predicted binarization, G is the GT, foreground values are 1 (logically true), background values are 0 (logically false), and the above boolean operations evaluate to 1 for true and 0 for false. FM has been used as part of all DIBCO competitions [36, 102, 114–120], as well as the PLM competition [16]. FM is used because there are often far more background pixels than foreground pixels, and it penalizes methods that make a disproportionate number of FP or FN errors.

Another metric used in all DIBCO competitions is the Peak Signal-to-Noise Ratio (PSNR), which is computed as

$$PSNR = 10\log(\frac{1}{MSE}) \qquad MSE = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(B(i,j) - G(i,j))^2 \tag{2.29}$$

Note that $MSE = FP + FN$ is simply a count of the number of errors made, which makes the PSNR metric monotonically increasing w.r.t. the traditional accuracy measure. This means that for a given image, PSNR and accuracy produce the same ranking order of algorithms.

Distance Reciprocal Distortion (DRD) has been used in DIBCO 2011-2018 and has been shown to correlate well with human perception of how bad binarization errors are [73]. It is computed by

$$DRD = \frac{1}{NUBN(G)}\sum_{ij} DRD_{ij}|B(i,j) - G(i,j)| \tag{2.30}$$

$$DRD_{ij} = \sum_{x=-2}^{2}\sum_{y=-2}^{2} W_{xy}|B(i+x, j+y) - G(i+x, j+y)| \tag{2.31}$$

where $NUBN(G)$ is the number of non-uniform 8x8 binary patches of G, and W is defined by $W_{00} = 0$, $\forall x \neq 0$, $y \neq 0$, $W_{xy} \propto \frac{1}{\sqrt{(x^2+y^2)}}$, and $\sum_{xy} W_{xy} = 1$. Lower DRD scores are better. From Eq. 2.30, we see that all wrongly predicted pixels contribute an error penalty according to Eq. 2.31. Eq. 2.31 examines the 5x5 window around the wrong pixel and assigns a penalty for every other wrong pixel in this window (but 0 penalty for itself). Thus if we have two predicted binarizations with an equal number of errors, DRD will assign the higher (worse) score to the prediction that has it's wrong predictions close together. Single isolated errors are not as noticeable to the human eye as concentrated groups of wrong pixels [73].

Pseudo F-measure

Ntirogiannis et al. [101] proposed a variant of FM, called pseudo-FMeasure (pFM) which is the harmonic mean of pseudo-Precision P_{ps} and pseudo-Recall R_{ps}. While Precision and Recall ignore the spatial locations of errors, P_{ps} and R_{ps} assign weights to each pixel based on the GT. These weights can be interpreted as how bad an error is at that location.

$$pFM = \frac{2P_{ps}R_{ps}}{P_{ps} + R_{ps}} \quad \begin{aligned} P_{ps} &= \frac{\sum_{ij} B(i,j)G(i,j)W_P(i,j)}{B(i,j)W_P(i,j)} \\ R_{ps} &= \frac{\sum_{ij} B(i,j)G(i,j)W_R(i,j)}{G(i,j)W_R(i,j)} \end{aligned} \tag{2.32}$$

where setting W_P and W_R to be uniform and non-zero reduces to the traditional FM as in Eq. 2.27. W_P assigns a greater penalty to FPs located near foreground regions. The rationale is that FP errors far from the foreground do not impede the readability of the text. FP pixels very close to the foreground-background boundary have lower weight due to the inherit ambiguity of localizing this boundary. Regions between foreground components also have high weight in order to penalize the joining of adjacent characters. W_R assigns the highest per-pixel error for FNs located on the skeleton of the foreground and on thinner strokes. Thus, binarizations that fragment characters yield a higher error than binarizations that predict thinner, but unfragmented strokes. Foreground pixels on the border are assigned a recall weight of 0, in order to account for single-pixel localization ambiguity of the boundary in the GT.

Ntirogiannis et al. [101] further compute a weighted measure of FN errors into the categories Fully Missed Text, Broken Text, and Partially Missed Text. Earlier similar definitions (uniform weights) were given in [100]. Fully Missed Text concerns the foreground components that are completely missed. Broken Text errors occur when FN errors break a

GT foreground component into 2 or more components. Partially Missed Text incorporates the rest of the errors, where a component is partially detected and not fragmented.

Likewise, the FP errors are categorized into False Alarms, Background Noise, Character Enlargement, and Character Merging. False Alarms are predicted components that do not overlap with any GT foreground component. Character Enlargement concerns FPs immediately around components that do not cause merging, while Character Merging errors are those pixels that cause merging of foreground components. Background Noise errors are similar to False Alarms, but are those pixels that are i) far away from GT components and are ii) part of a predicted component that overlaps a GT component.

While such a detailed break-down of errors is certainly useful for comparative analysis of algorithms, these or similar numerical measures are rarely reported in the literature. This may be because there is no public code available for evaluating this error breakdown like there is for pFM, FM, PSNR, and DRD. Djema et al. [29] also advocate that error analysis should be done based on the types of degradations present in the images.

In an unfortunate naming collision, a different metric also called pseudo F-measure [100] was previously used in the 2010 and 2012 DIBCO competitions, while the newer pseudo F-measure [101] is now prefered and has been used in DIBCO 2013-2018. The old pseudo F-measure [100] is like the traditional FM, but when computing recall, only the skeleton of the GT is used. In contrast, the new pseudo F-measure [101] evaluates recall with all non-border foreground pixels, increasing the penalty for errors based on distance to the GT skeleton.

2.8 Evaluation without Pixel Ground Truth

Though most recently proposed methods perform evaluation using GT (Section 2.7), there has been some discussion on whether this is a useful form of evaluation. Given the problems with defining objective GT at the pixel level (Section 2.8.1), some alternative evaluation methods have been proposed that use blind metrics (Section 2.8.2), including consensus based metrics (Section 2.8.2), and end-to-end systems (Section 2.8.3).

2.8.1 Problems with Pixel GT

While the DIBCO framework for ground truth creation allows for efficient user interaction, the heavy use of automation can introduce bias towards the algorithms used for automation. For example, Howe [45] noted that the disproportionate number of errors observed on the north-west side of CCs could be an artifact of the GT construction process. Though [94] uses less automation, there still exists the possibility of bias toward the algorithm used in the initial binarization.

For the ICHFR 2016 PLM competition [16] over the AMADI_LONTAR dataset [56], two sets of binarization GT were released, created by different annotators. The agreement F-measure between the two annotators is 62.40%. In [55], an analysis of annotator variability is done under the scenario where the annotators manually draw all character skeletons, i.e. no initial automatic binarization is performed. They report an average agreement of 59.56% FM, which is lower than the top two binarization methods in the competition [16] (68% and 63% FM).

Smith [143] conducted a study where 6 students were asked to annotate a single DIBCO image using the PixLabler [134] tool. The average pairwise FM for these 6 *ground truths* was 84.9%. Similarly, a single student produced annotations for all 14 DIBCO 2009 images [36] and achieved an average FM score of 89.3% w.r.t the published GT. This is lower than the 91.24% FM achieved by the competition winner [36] and lower than the 94.68% reported by Howe [46].

In a separate study, Smith and An [144] experimented with bias versions of GT (eroded, original, dilated) for training and evaluating a learning based binarization method. Unsurprisingly, the highest evaluation scores were achieved when the training GT bias matched the evaluation GT bias. However, this highlights the problem of ranking algorithms based on a single inherently biased GT.

2.8.2 Blind Metrics

Blind metrics evaluate performance using no external GT information at test time. Though any such metric can be turned into a binarization algorithm by performing optimization with the blind metric as the objective function [124], they are still useful measures, particularly for parameter tuning.

In [80], GT is constructed automatically and used to choose binarization methods and tune parameters. Stommel and Frieder [148] trained an SVM classifier to predict the global legibility of binary images using a labeled training set of binary images. Afterwards, they pick a global threshold that optimizes the predicted legibility of a test image. Ramírez-Ortegón et al. [124] proposed a statistical measure, based on the assumption that foreground and background regions are log-normally distributed, that was shown to correlate well with OCR accuracy.

Shaus et al. [141] proposed a number of blind metrics, many of which are adapted from older global thresholding techniques (e.g. Otsu [105]), based on the separation of foreground and background clusters of pixels. However, turning a binarization algorithm into a metric leads to that binarization algorithm producing the optimal result for that metric, regardless of how well it actually correlated with human perception of binarization performance. For example, one proposed metric is the PSNR of the binary image w.r.t. the input grayscale image. While intuitively lighter pixels should be assigned to background and vice versa, this metric is trivially optimized by choosing a global threshold of $T = 128$ for 8-bit grayscale images.

Consensus Methods

Fedorchuk and Lamiroy [31] proposed a method for evaluating binarizations based on how the predicted binarization agrees with the output of other algorithms in the absence of ground truth. Essentially, the binary GT image is replaced with a probabilistic image, where the probabilities are the percentage of algorithms that consider that pixel as foreground. They found that the probabilistic versions of FM and PSNR had an average correlation coefficient of 85% with the traditional non-probabilistic FM and PSNR measures on the DIBCO datasets. However, the top performing algorithms on the DIBCO data achieve an FM greater than 90%, so these proposed consensus measures could not ascertain if a new algorithm performs better than the current state-of-the-art.

In a similar vein, paired ranking based on agreement with a third, better than random, model has been proposed [32]. Such a scheme works if the third model makes unbiased, random errors, but this assumption would never be satisfied in practice. Different choices of a third model would yield different biases and hence different rankings.

Overall, consensus based evaluation has similar disadvantages as other blind evaluations, specifically bias introduced by the choice of models to compare with. The majority vote of the algorithms used for consensus trivially optimizes the proposed consensus metrics.

2.8.3 End-to-End Metrics

Most often binarization algorithms are intended to be used as a preprocessing component in a larger system that also performs other tasks, such as layout analysis or text recognition. Thus it is logical to compare binarization algorithms based on how they influence the performance of the whole system. Before annotated images were introduced in 2009 by the DIBCO competition, this was the primary way to numerically compare binarization algorithms using real data.

As such, many works also evaluate the quality of their binarization method by measuring the character and word error rates of a third-party OCR engine (e.g. ABBYY[1]) run over their binary images [20, 35, 59, 66, 86, 96, 121, 154]. While this introduces the bias of the OCR engine (it may be more forgiving to some types of errors), one may argue that, in the context of real-world applications, end-to-end performance is more important than intermediate binarization quality.

However, the comparative ranking of algorithms depends on the choice of end-to-end system. To reduce system bias, perhaps evaluation should be done with multiple state-of-the-art end-to-end systems, where binarization method ranking is based on the maximum performance achieved with any systems. Though such evaluation would be taxing on academics that have little resources to implement full systems, recent work in building DIA systems as web services [163] may facilitate such an approach.

We also note that there are no standard datasets designed for evaluating binarization algorithms using OCR performance. The DIBCO data is not annotated with transcriptions and is multi-lingual. All papers surveyed in this review that used OCR to evaluate binarization quality used a distinct set of images (some synthetic), which hinders comparative evaluation. If this method of evaluation is to be used, there needs to be a dataset designed for this purpose.

2.9 Conclusion

In this work, we have reviewed the recent advances in the field of historical document binarization. For many years, methods based on image processing (e.g. [151]) and CRFs (e.g. [46]) produced state-of-the-art results, especially when combined with preprocessing, post-processing, and parameter tuning. Very recently, pixel classification approaches using deep learning models are the state-of-the-art. Though deep models are accurate, they are slow and require GPU hardware to obtain reasonable speed. Some researchers have focused

[1] https://www.abbyy.com/en-us/finereader/

on making existing binarization approaches efficient enough to process large archives, and more research is needed in this area.

Recent years have also seen an expanded interest in binarizing other document domains such as PLMs. There continues to be debate on the proper way to evaluate binarization algorithms, given that humans do not always agree on the pixel-level boundaries between foreground and background. However, the majority of published research continues to evaluate at the pixel level using the DIBCO provided annotations.

Future work in the area might explore how to integrate some of the best practices from image processing techniques with deep learning models or vice versa. For example, incorporating preprocessing and thresholding into the learning model or using deep models to replace many of the parameters or fixed constants in traditional methods. A large dataset suitable for data-hungry deep models with high-quality, low-bias pixel annotations also remains elusive. While the pseudo F-measure metric [101] partly addresses some concerns with using pixel GT, less biased evaluation measures are needed. One possible avenue forward is to create multiple reproducible end-to-end document processing pipelines (binarization, layout analysis, text recognition, retrieval, writer ID, etc.) for different tasks and domains. To prevent bias towards particular downstream implementations, multiple implementations could be tried and performance recorded for the best combination of downstream implementations. Then binarization algorithms could be comparatively evaluated in multiple real contexts and domains and give a more comprehensive view of algorithm performance.

References

[1] Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki. Deepdocclassifier: Document classification with deep convolutional neural network. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE, 2015.

[2] Ehsan Ahmadi, Zohreh Azimifar, Maryam Shams, Mahmoud Famouri, and Mohammad Javad Shafiee. Document image binarization using a discriminative structural classifier. *Pattern Recognition Letters*, 63:36–42, Elsevier, 2015.

[3] Marcos Almeida, Rafael Dueire Lins, Rodrigo Bernardino, Darlisson Jesus, and Bruno Lima. A new binarization algorithm for historical documents. *Journal of Imaging*, 4 (2), Multidisciplinary Digital Publishing Institute, 2018.

[4] A.W.A. Arruda and Carlos A.B. Mello. Binarization of degraded document images based on combination of contrast images. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 615–620. IEEE, 2014.

[5] Seyed Morteza Ayatollahi and Hossein Ziaei Nafchi. Persian heritage image binarization competition (PHIBC 2012). In *Iranian Conference on Pattern Recognition and Image Analysis (PRIA)*, pages 1–4. IEEE, 2013.

[6] Kalyan Ram Ayyalasomayajula and Anders Brun. Document binarization using topological clustering guided laplacian energy segmentation. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 523–528. IEEE, 2014.

[7] Kalyan Ram Ayyalasomayajula, Filip Malmberg, and Anders Brun. PDNet: Semantic segmentation integrated with a primal-dual network for document binarization. *Pattern Recognition Letters*, pages 52–60, Elsevier, 2018.

[8] Soumen Bag and Partha Bhowmick. Adaptive–interpolative binarization with stroke preservation for restoration of faint characters in degraded documents. *Journal of Visual Communication and Image Representation*, 31:266–281, Elsevier, 2015.

[9] Bilal Bataineh, Siti Norul Huda Sheikh Abdullah, and Khairuddin Omar. An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. *Pattern Recognition Letters*, 32(14):1805–1813, Elsevier, 2011.

[10] Bilal Bataineh, Siti Norul Huda Sheikh Abdullah, and Khairuddin Omar. Adaptive binarization method for degraded document images based on surface contrast variation. *Pattern Analysis and Applications*, 20(3):639–652, Springer, 2017.

[11] Rajesh K. Bawa and Ganesh K. Sethi. A review on binarization algorithms for camera based natural scene images. In *International Conference on Advances in Computing, Communications and Informatics*, pages 873–878. ACM, 2012.

[12] Costin-Anton Boiangiu, Alexandra Olteanu, Alexandru Stefanescu, Daniel Rosner, Nicolae Tapus, and Mugurel Ionut Andreica. Local thresholding algorithm based on variable window size statistics. In *International Conference on Control Systems and Computer Science (CSCS)*, volume 2, pages 647–652, 2011.

[13] Su Bolan, Lu Shijian, and Chew Lim Tan. A self-training learning document binarization framework. In *International Conference on Pattern Recognition (ICPR)*, pages 3187–3190. IEEE, 2010.

[14] Manuel Bouillon, Rolf Ingold, and Marcus Liwicki. Grayification: a meaningful grayscale conversion to improve handwritten historical documents analysis. *Pattern Recognition Letters*, Elsevier, 2018.

[15] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, IEEE, 2004.

[16] Jean-Christophe Burie, Mickaël Coustaty, Setiawan Hadi, Made Windu Antara Kesiman, Jean-Marc Ogier, Erick Paulus, Kimheng Sok, I. Made Gede Sunarya, and Dona Valy. ICFHR 2016 competition on the analysis of handwritten text in images of balinese palm leaf manuscripts. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 596–601, 2016.

[17] Jean-Christophe Burie, Made Windu Antara Kesiman, Dona Valy, Erick Paulus, Mira Suryani, Setiawan Hadi, Michel Verleysen, Sophea Chhun, and Jean-Marc Ogier. ICFHR2018 competition on document image analysis tasks for southeast asian palm leaf manuscripts. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2018.

[18] Jorge Calvo-Zaragoza and Antonio-Javier Gallego. A selectional auto-encoder approach for document image binarization. *Pattern Recognition*, 86:37–47, Elsevier, 2019.

[19] John Canny. A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, IEEE, 1986.

[20] Tanushyam Chattopadhyay, V. Ramu Reddy, and Utpal Garain. Automatic selection of binarization method for robust OCR. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1170–1174. IEEE, 2013.

[21] Prashali Chaudhary and B.S. Saini. An effective and robust technique for the binarization of degraded document images. *International Journal of Research in Engineering and Technology (IJRET)*, 3(06):2319–1163, Citeseer, 2014.

[22] Kai Chen, Mathias Seuret, Jean Hennebert, and Rolf Ingold. Convolutional neural networks for page segmentation of historical document images. In *International Con-*

ference on Document Analysis and Recognition (ICDAR), volume 1, pages 965–970. IEEE, 2017.

[23] Xin Chen, Liang Lin, and Yuefang Gao. Parallel nonparametric binarization for degraded document images. *Neurocomputing*, 189:43–52, Elsevier, 2016.

[24] Mohamed Cheriet, Reza Farrahi Moghaddam, and Rachid Hedjam. A learning framework for the optimization and automation of document binarization methods. *Computer Vision and Image Understanding*, 117(3):269–280, Elsevier, 2013.

[25] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[26] Yung-Hsiang Chiu, Kuo-Liang Chung, Wei-Ning Yang, Yong-Huai Huang, and Chi-Huang Liao. Parameter-free based two-stage method for binarizing degraded document images. *Pattern Recognition*, 45(12):4250–4262, Elsevier, 2012.

[27] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[28] Fanbo Deng, Zheng Wu, Zheng Lu, and Michael S. Brown. BinarizationShop: A user-assisted software suite for converting old documents to black-and-white. In *Joint Conference on Digital libraries*, pages 255–258. ACM, 2010.

[29] Amina Djema, Youcef Chibani, Abdenour Sehad, and E.T. Tahir Zemouri. Blind versus unblind performance evaluation of binarization methods. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 511–515. IEEE, 2015.

[30] Mohamed Abd Elfattah, Aboul Ella Hassanien, Sherihan Abuelenin, and Siddhartha Bhattacharyya. Multi-verse optimization clustering algorithm for binarization of hand-

written documents. In *Recent Trends in Signal and Image Processing*, pages 165–175. Springer, 2019.

[31] Maksym Fedorchuk and Bart Lamiroy. Statistic metrics for evaluation of binary classifiers without ground-truth. In *Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pages 1066–1071. IEEE, 2017.

[32] Maksym Fedorchuk and Bart Lamiroy. Binary classifier evaluation without ground truth. In *International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–6. IEEE, 2017.

[33] Rafael Felix, Leandro Augusto da Silva, and Leandro Nunes de Castro. Thresholding the courtesy amount of brazilian bank checks using a local methodology. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 213–221. Springer, 2015.

[34] Basura Fernando, Sezer Karaoglu, and Alain Trémeau. Extreme value theory based text binarization in documents and natural scenes. In *International Conference on Machine Vision (ICMV)*, pages 144–151. IEEE, 2010.

[35] Basilios Gatos, Ioannis Pratikakis, and Stavros J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39(3):317–327, Elsevier, 2006.

[36] Basilios Gatos, Konstantinos Ntirogiannis, and Ioannis Pratikakis. ICDAR 2009 document image binarization contest (DIBCO 2009). In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 9, pages 1375–1382. IEEE, 2009.

[37] Basilios Gatos, Konstantinos Ntirogiannis, and Ioannis Pratikakis. DIBCO 2009: Document image binarization contest. *International Journal on Document Analysis and Recognition*, 14(1):35–44, Springer, 2011.

[38] Zineb Hadjadj, Abdelkrim Meziane, Yazid Cherfa, Mohamed Cheriet, and Insaf Setitra. ISauvola: Improved sauvola's algorithm for document image binarization. In *Image Analysis and Recognition*, pages 737–745. Springer, 2016.

[39] Zineb Hadjadj, Mohamed Cheriet, Abdelkrim Meziane, and Yazid Cherfa. A new efficient binarization method: Application to degraded historical document images. *Signal, Image and Video Processing*, 11(6):1155–1162, Springer, 2017.

[40] Hatem Hamza, Eddie Smigiel, and E. Belaid. Neural based binarization techniques. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 317–321. IEEE, 2005.

[41] Aboul Ella Hassanien, Mohamed Abd Elfattah, Sherihan Aboulenin, Gerald Schaefer, Shao Ying Zhu, and Iakov Korovin. Historic handwritten manuscript binarisation using whale optimisation. In *International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3842–3846. IEEE, 2016.

[42] David Hebert, Stephane Nicolas, and Thierry Paquet. Discrete CRF based combination framework for document image binarization. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1165–1169. IEEE, 2013.

[43] Rachid Hedjam, Reza Farrahi Moghaddam, and Mohamed Cheriet. A spatially adaptive statistical method for the binarization of historical manuscripts and degraded document images. *Pattern Recognition*, 44(9):2184–2196, Elsevier, 2011.

[44] Rachid Hedjam, Hossein Ziaei Nafchi, Margaret Kalacska, and Mohamed Cheriet. Influence of color-to-gray conversion on the performance of document image binarization: Toward a novel optimization problem. *Transactions on Image Processing*, 24(11):3637–3651, IEEE, 2015.

[45] Nicholas R. Howe. A laplacian energy for document binarization. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 6–10. IEEE, 2011.

[46] Nicholas R. Howe. Document binarization with automatic parameter tuning. *International Journal on Document Analysis and Recognition*, 16(3):247–258, Springer, 2013.

[47] Saad M. Ismail, Siti Norul Huda Sheikh Abdullah, and Fariza Fauzi. Statistical binarization techniques for document image analysis. *Journal of Computer Science*, 14 (1):23–36, Science Publications, 2018.

[48] J. Jennifer Ranjani. Bi-level thresholding for binarisation of handwritten and printed documents. *IET Computer Vision*, 10, IET, 2015.

[49] Fuxi Jia, Cunzhao Shi, Kun He, Chunheng Wang, and Baihua Xiao. Degraded document image binarization using structural symmetry of strokes. *Pattern Recognition*, 74:225–240, Elsevier, 2018.

[50] Nicholas Journet, Muriel Visani, Boris Mansencal, Kieu Van-Cuong, and Antoine Billy. DocCreator: A new software for creating synthetic ground-truthed document images. *Journal of Imaging*, 3(4):62, Multidisciplinary Digital Publishing Institute, 2017.

[51] Fauziah Kasmin, Azizi Abdullah, and Anton Satria Prabuwono. Ensemble of steerable local neighbourhood grey-level information for binarization. *Pattern Recognition Letters*, 98:8–15, Elsevier, 2017.

[52] Abderrahmane Kefali, Toufik Sari, and Mokhtar Sellami. Evaluation of several binarization techniques for old Arabic documents images. In *International Symposium on Modeling and Implementing Complex Systems*, volume 1, pages 88–99. Springer, 2010.

[53] Abderrahmane Kefali, Toufik Sari, and Halima Bahi. Foreground-background separation by feed-forward neural networks in old manuscripts. *Informatica*, 38(4), 2014.

[54] Made Windu Antara Kesiman, Sophea Prum, Jean-Christophe Burie, and Jean-Marc Ogier. An initial study on the construction of ground truth binarized images of

ancient palm leaf manuscripts. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 656–660. IEEE, 2015.

[55] Made Windu Antara Kesiman, Sophea Prum, I. Made Gede Sunarya, Jean-Christophe Burie, and Jean-Marc Ogier. An analysis of ground truth binarized image variability of palm leaf manuscripts. In *International Conference on Image Processing, Theory, Tools and Applications (IPTA)*, pages 229–233. IEEE, 2015.

[56] Made Windu Antara Kesiman, Jean-Christophe Burie, Gusti Ngurah Made Agus Wibawantara, I. Made Gede Sunarya, and Jean-Marc Ogier. AMADI_LontarSet: the first handwritten balinese palm leaf manuscripts dataset. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 168–173. IEEE, 2016.

[57] Krisda Khankasikam. Restoration of degraded historical document image: An adaptive multilayer-information binarization technique. *Journal of Information Science and Engineering*, 30(5):1321–1338, 2014.

[58] Mehdi Khitas, Lahcene Ziet, and Saad Bouguezel. Improved degraded document image binarization using median filter for background estimation. *Elektronika ir Elektrotechnika*, 24(3):82–87, 2018.

[59] Khurram Khurshid, Imran Siddiqi, Claudie Faure, and Nicole Vincent. Comparison of niblack inspired binarization methods for ancient documents. In *Document Recognition and Retrieval XVI*, volume 7247. International Society for Optics and Photonics, 2009.

[60] Netanel Kligler, Sagi Katz, and Ayellet Tal. Document enhancement using visibility detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2374–2382. IEEE, 2018.

[61] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *Transactions on Pattern Analysis and Machine Intelligence*, 26(2): 147–159, IEEE, 2004.

[62] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117. MIT Press, 2011.

[63] Jung Gap Kuk and Nam Ik Cho. Feature based binarization of document images degraded by uneven light condition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 748–752. IEEE, 2009.

[64] T. Hoang Ngan Le, Tien D. Bui, and Ching Y. Suen. Ternary entropy-based binarization of degraded document images using morphological operators. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 114–118. IEEE, 2011.

[65] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, IEEE, 1998.

[66] Thibault Lelore and Frédéric Bouchara. Document image binarisation using markov field model. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 551–555. IEEE, 2009.

[67] Thibault Lelore and Frederic Bouchara. FAIR: A fast algorithm for document image restoration. *Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2039–2048, IEEE, 2013.

[68] Yunfeng Liang, Zhiping Lin, Lei Sun, and Jiuwen Cao. Document image binarization via optimized hybrid thresholding. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[69] Laurence Likforman-Sulem, Jérôme Darbon, and Elisa H. Barney Smith. Enhancement of historical printed document images by combining total variation regularization and non-local means filtering. *Image and Vision Computing*, 29(5):351–363, Elsevier, 2011.

[70] Rafael Dueire Lins, Marcos Martins de Almeida, Rodrigo Barros Bernardino, Darlisson Jesus, and José Mário Oliveira. Assessing binarization techniques for document images. In *Symposium on Document Engineering (DocEng)*, pages 183–192. ACM, 2017.

[71] Ning Liu, Dongxiang Zhang, Xing Xu, Wenju Liu, Dengfeng Ke, Long Guo, Shengkun Shi, Hui Liu, and Lijiang Chen. An iterative refinement framework for image document binarization with Bhattacharyya similarity measure. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 93–98. IEEE, 2017.

[72] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 3431–3440. IEEE, 2015.

[73] Haiping Lu, Alex C. Kot, and Yun Q. Shi. Distance-reciprocal distortion measure for binary document images. *Signal Processing Letters*, 11(2):228–231, IEEE, 2004.

[74] Shijian Lu, Bolan Su, and Chew Lim Tan. Document image binarization using background estimation and stroke edges. *International Journal on Document Analysis and Recognition*, 13(4):303–314, Springer, 2010.

[75] Zheng Lu, Zheng Wu, and Michael S. Brown. Interactive degraded document binarization: An example (and case) for interactive computer vision. In *Workshop on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2009.

[76] Zheng Lu, Zheng Wu, and Michael S. Brown. Directed assistance for ink-bleed reduction in old documents. In *Computer Vision and Pattern Recognition (CVPR)*, pages 88–95. IEEE, 2009.

[77] Rafael G. Mesquita, Carlos A. B. Mello, and L. H. E. V. Almeida. A new thresholding algorithm for document images based on the perception of objects by distance. *Integrated Computer-Aided Engineering*, 21(2):133–146, IOS Press, 2014.

[78] Rafael G. Mesquita, Ricardo M. A. Silva, Carlos A. B. Mello, and Péricles B. C. Miranda. Parameter tuning for document image binarization using a racing algorithm. *Expert Systems with Applications*, 42(5):2593–2603, Elsevier, 2015.

[79] Ines Ben Messaoud, Haikal El Abed, Hamid Amiri, and Volker Märgner. New method for the selection of binarization parameters based on noise features of historical documents. In *Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, page 1. ACM, 2011.

[80] Ines Ben Messaoud, Haikal El Abed, Volker Märgner, and Hamid Amiri. A design of a preprocessing framework for large database of historical documents. In *Workshop on Historical Document Imaging and Processing (HIP)*, pages 177–183. ACM, 2011.

[81] Ines Ben Messaoud, Hamid Amiri, Haikal El Abed, and Volker Margner. Document preprocessing system-automatic selection of binarization. In *Document Analysis Systems (DAS)*, pages 85–89. IEEE, 2012.

[82] Ines Ben Messaoud, Hamid Amiri, Haikal El Abed, and Volker Märgner. Region based local binarization approach for handwritten ancient documents. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 633–638. IEEE, 2012.

[83] Anand Mishra, Karteek Alahari, and C. V. Jawahar. Unsupervised refinement of color and stroke features for text binarization. *International Journal on Document Analysis and Recognition*, 20(2):105–121, Springer, 2017.

[84] Nikolaos Mitianoudis and Nikolaos Papamarkos. Local co-occurrence and contrast mapping for document image binarization. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 609–614. IEEE, 2014.

[85] Nikolaos Mitianoudis and Nikolaos Papamarkos. Document image binarization using local features and gaussian mixture modeling. *Image and Vision Computing*, 38:33–51, Elsevier, 2015.

[86] Reza Farrahi Moghaddam and Mohamed Cheriet. A multi-scale framework for adaptive binarization of degraded document images. *Pattern Recognition*, 43(6):2186–2198, Elsevier, 2010.

[87] Reza Farrahi Moghaddam and Mohamed Cheriet. AdOtsu: An adaptive and parameterless generalization of Otsu's method for document image binarization. *Pattern Recognition*, 45(6):2419–2431, Elsevier, 2012.

[88] Reza Farrahi Moghaddam, Fereydoun Farrahi Moghaddam, and Mohamed Cheriet. Unsupervised ensemble of experts (EOE) framework for automatic binarization of document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 703–707. IEEE, 2013.

[89] Ayatullah Faruk Mollah, Subhadip Basu, and Mita Nasipuri. Computationally efficient implementation of convolution-based locally adaptive binarization techniques. In *Wireless Networks and Computational Intelligence*, pages 159–168. Springer, 2012.

[90] Prachi K More and D. D. Dighe. A review on document image binarization technique for degraded document images. *International Research Journal of Engineering and Technology*, pages 1132–1138, 2016.

[91] Wan Azani Mustafa, Hairy Aziz, Wan Khairunizam, Zunaidi Ibrahim, A. B. Shahriman, and Zuradzman M. Razlan. Review of different binarization approaches on degraded document images. In *International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, pages 1–8. IEEE, 2018.

[92] Hossein Ziaei Nafchi and Hamidreza Rashidy Kanan. A phase congruency based document binarization. In *International Conference on Image and Signal Processing*, pages 113–121. Springer, 2012.

[93] Hossein Ziaei Nafchi, Reza Farrahi Moghaddam, and Mohamed Cheriet. Historical document binarization based on phase information of images. In *Asian Conference on Computer Vision (ACCV)*, pages 1–12. Springer, 2012.

[94] Hossein Ziaei Nafchi, Seyed Morteza Ayatollahi, Reza Farrahi Moghaddam, and Mohamed Cheriet. An efficient ground truthing tool for binarization of historical manuscripts. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 807–811. IEEE, 2013.

[95] Hossein Ziaei Nafchi, Reza Farrahi Moghaddam, and Mohamed Cheriet. Phase-based binarization of ancient document images: Model and applications. *Transactions on Image Processing*, 23(7):2916–2930, IEEE, 2014.

[96] Jayanthi Natarajan and Indu Sreedevi. Enhancement of ancient manuscript images by log based binarization technique. *AEU-International Journal of Electronics and Communications*, 75:15–22, Elsevier, 2017.

[97] Wayne Niblack. *An introduction to digital image processing.* Strandberg Publishing Company, 1985.

[98] Anguelos Nicolaou, Marcus Liwicki, and Rolf Ingolf. Investigating the power of integral histograms for document images, a binarization case study, 2014. `http://nicolaou.homouniversalis.org/assets/lof/nicolaou2013binarization_lowres.pdf`.

[99] Oliver Nina, Bryan Morse, and William Barrett. A recursive Otsu thresholding method for scanned document binarization. In *Workshop on Applications of Computer Vision (WACV)*, pages 307–314. IEEE, 2011.

[100] Konstantinos Ntirogiannis, Basilios Gatos, and Ioannis Pratikakis. An objective evaluation methodology for document image binarization techniques. In *Document Analysis Systems (DAS)*, pages 217–224. IEEE, 2008.

[101] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. Performance evaluation methodology for historical document image binarization. *Transactions on Image Processing*, 22(2):595–609, IEEE, 2013.

[102] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. ICFHR2014 competition on handwritten document image binarization (H-DIBCO 2014). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 809–813. IEEE, 2014.

[103] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. A combined approach for the binarization of handwritten document images. *Pattern Recognition Letters*, 35: 3–15, Elsevier, 2014.

[104] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, IEEE, 2002.

[105] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, IEEE, 1979.

[106] Naouel Ouafek and Mohamed-Khireddine Kholladi. A binarization method for degraded document image using artificial neural network and interpolation inpainting. In *International Conference on Optimization and Applications (ICOA)*, pages 1–5. IEEE, 2018.

[107] Roberto Paredes, Ergina Kavallieratou, and Rafael Dueire Lins. ICFHR 2010 contest: Quantitative evaluation of binarization algorithms. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 733–736. IEEE, 2010.

[108] Jon Parker, Ophir Frieder, and Gideon Frieder. Robust binarization of degraded document images using heuristics. In *Document Recognition and Retrieval XXI*, volume 9021. International Society for Optics and Photonics, 2014.

[109] J. Pastor-Pellicer, S. España-Boquera, F. Zamora-Martínez, M. Zeshan Afzal, and Maria Jose Castro-Bleda. Insights on the use of convolutional neural networks for document image binarization. In *International Work-Conference on Artificial Neural Networks*, pages 115–126. Springer, 2015.

[110] Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España-Boquera, and María José Castro-Bleda. F-measure as the error function to train neural networks. In *International Work-Conference on Artificial Neural Networks*, pages 376–384. Springer, 2013.

[111] Xujun Peng, Srirangaraj Setlur, Venu Govindaraju, and Ramachandrula Sitaram. Markov random field based binarization for hand-held devices captured document images. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 71–76. ACM, 2010.

[112] Xujun Peng, Huaigu Cao, and Prem Natarajan. Using convolutional encoder-decoder for document image binarization. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 708–713. IEEE, 2017.

[113] Benjamin Perret, Sébastien Lefèvre, Christophe Collet, and Eric Slezak. From hyperconnections to hypercomponent tree: Application to document image binarization. In *Workshop on Applications of Discrete Geometry and Mathematical Morphology (WADGMM)*, page 62. IAPR, 2010.

[114] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. H-DIBCO 2010 - handwritten document image binarization competition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 727–732. IEEE, 2010.

[115] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2011 document image binarization contest (DIBCO 2011). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1506–1510. IEEE, 2011.

[116] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. ICFHR 2012 competition on handwritten document image binarization (H-DIBCO 2012). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 817–822. IEEE, 2012.

[117] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2013 document image binarization contest (DIBCO 2013). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1471–1476. IEEE, 2013.

[118] Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. ICFHR2016 handwritten document image binarization contest (H-DIBCO 2016). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 619–623. IEEE, 2016.

[119] Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. ICDAR2017 competition on document image binarization (DIBCO 2017). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1395–1403. IEEE, 2017.

[120] Ioannis Pratikakis, Konstantinos Zagoris, Panagiotis Kaddas, and Basilis Gatos. ICFHR 2018 competition on handwritten document image binarization contest (H-DIBCO 2018). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 1–1. IEEE, 2018.

[121] Juliano C. B. Rabelo, Cleber Zanchettin, Carlos A. B. Mello, and Byron L. D. Bezerra. A multi-layer perceptron approach to threshold documents with complex background. In

International Conference on Systems, Man, and Cybernetics (SMC), pages 2523–2530. IEEE, 2011.

[122] Marte A. Ramírez-Ortegón, Ernesto Tapia, Lilia L. Ramírez-Ramírez, Raúl Rojas, and Erik Cuevas. Transition pixel: A concept for binarization based on edge detection and gray-intensity histograms. *Pattern Recognition*, 43(4):1233–1243, Elsevier, 2010.

[123] Marte A. Ramírez-Ortegón, Ernesto Tapia, Raúl Rojas, and Erik Cuevas. Transition thresholds and transition operators for binarization and edge detection. *Pattern Recognition*, 43(10):3243–3254, Elsevier, 2010.

[124] Marte A. Ramírez-Ortegón, Edgar A. Duéñez-Guzmán, Raúl Rojas, and Erik Cuevas. Unsupervised measures for parameter selection of binarization algorithms. *Pattern Recognition*, 44(3):491–502, Elsevier, 2011.

[125] Marte A. Ramírez-Ortegón, Volker Märgner, Erik Cuevas, and Raúl Rojas. An optimization for binarization methods by removing binary artifacts. *Pattern Recognition Letters*, 34(11):1299–1306, Elsevier, 2013.

[126] Marte A. Ramírez-Ortegón, Lilia L. Ramírez-Ramírez, Volker Märgner, Ines Ben Messaoud, Erik Cuevas, and Raúl Rojas. An analysis of the transition proportion for binarization in handwritten historical documents. *Pattern Recognition*, 47(8):2635–2651, Elsevier, 2014.

[127] Marte A. Ramírez-Ortegón, Lilia L. Ramírez-Ramírez, Ines Ben Messaoud, Volker Märgner, Erik Cuevas, and Raúl Rojas. A model for the gray-intensity distribution of historical handwritten documents and its application for binarization. *International Journal on Document Analysis and Recognition*, 17(2):139–160, Springer, 2014.

[128] David Rivest-Hénault, Reza Farrahi Moghaddam, and Mohamed Cheriet. A local linear level set method for the binarization of degraded historical document images.

International Journal on Document Analysis and Recognition, 15(2):101–124, Springer, 2012.

[129] Edward Roe and Carlos A. B. Mello. Thresholding color images of historical documents with preservation of the visual quality of graphical elements. *Integrated Computer-Aided Engineering*, pages 1–12, IOS Press, 2018.

[130] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[131] Róisín Rowley-Brooke, François Pitié, and Anil Kokaram. A ground truth bleed-through document image database. In *International Conference on Theory and Practice of Digital Libraries*, pages 185–196. Springer, 2012.

[132] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, Springer, 2015.

[133] Khairun Saddami, Khairul Munadi, Sayed Muchallil, and Fitri Arnia. Improved thresholding method for enhancing jawi binarization performance. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1108–1113. IEEE, 2017.

[134] Eric Saund, Jing Lin, and Prateek Sarkar. Pixlabeler: User interface for pixel-level labeling of elements in document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 646–650. IEEE, 2009.

[135] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, Elsevier, 2000.

[136] Abdenour Sehad, Youcef Chibani, Rachid Hedjam, and Mohamed Cheriet. Gabor filter-based texture for ancient degraded document image binarization. *Pattern Analysis and Applications*, pages 1–22, Springer, 2018.

[137] Boran Şekeroğlu and Adnan Khashman. Performance evaluation of binarization methods for document images. In *International Conference on Advances in Image Processing*, pages 96–102. ACM, 2017.

[138] Mathias Seuret, Kai Chen, Nicole Eichenbergery, Marcus Liwicki, and Rolf Ingold. Gradient-domain degradations for improving historical documents images layout analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1006–1010. IEEE, 2015.

[139] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–166, International Society for Optics and Photonics, 2004.

[140] Faisal Shafait, Joost Van Beusekom, Daniel Keysers, and Thomas M. Breuel. Document cleanup using page frame detection. *International Journal on Document Analysis and Recognition*, 11(2):81–96, Springer, 2008.

[141] Arie Shaus, Barak Sober, Eli Turkel, and Eli Piasetzky. Beyond the ground truth: Alternative quality measures of document binarizations. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 495–500. IEEE, 2016.

[142] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962. IEEE, 2003.

[143] Elisa H. Barney Smith. An analysis of binarization ground truthing. In *Document Analysis Systems (DAS)*, pages 27–34. ACM, 2010.

[144] Elisa H. Barney Smith and Chang An. Effect of "ground truth" on image binarization. In *Document Analysis Systems (DAS)*, pages 250–254. IEEE, 2012.

[145] Elisa H. Barney Smith, Laurence Likforman-Sulem, and Jérôme Darbon. Effect of pre-processing on binarization. In *Document Recognition and Retrieval XVII*, volume 7534. International Society for Optics and Photonics, 2010.

[146] Vavilis Sokratis and Ergina Kavallieratou. A tool for tuning binarization techniques. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1–5. IEEE, 2011.

[147] Pyrrhos Stathis, Ergina Kavallieratou, and Nikos Papamarkos. An evaluation technique for binarization algorithms. *Journal of Universal Computer Science*, 14(18):3011–3030, 2008.

[148] Martin Stommel and Gideon Frieder. Automatic estimation of the legibility of binarised historic documents for unsupervised parameter tuning. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 104–108. IEEE, 2011.

[149] Bolan Su, Shijian Lu, and Chew Lim Tan. Combination of document image binarization techniques. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 22–26. IEEE, 2011.

[150] Bolan Su, Shijian Lu, and Chew Lim Tan. A learning framework for degraded document image binarization using markov random field. In *International Conference on Pattern Recognition (ICPR)*, pages 3200–3203. IEEE, 2012.

[151] Bolan Su, Shijian Lu, and Chew Lim Tan. Robust document image binarization technique for degraded document images. *Transactions on Image Processing*, 22(4): 1408–1417, IEEE, 2013.

[152] Bolan Su, Shuangxuan Tian, Shijian Lu, Thien Anh Dinh, and Chew Lim Tan. Self learning classification for degraded document images by sparse representation. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 155–159. IEEE, 2013.

[153] Chris Tensmeyer and Tony Martinez. Document image binarization with fully convolutional neural networks. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 99–104. IEEE, 2017.

[154] Morteza Valizadeh and Ehsanollah Kabir. An adaptive water flow model for binarization of degraded document images. *International Journal on Document Analysis and Recognition*, 16(2):165–176, Springer, 2013.

[155] Ekta Vats, Anders Hast, and Prashant Singh. Automatic document image binarization using bayesian optimization. In *Workshop on Historical Document Imaging and Processing (HIP)*, pages 89–94. ACM, 2017.

[156] Garret D. Vo and Chiwoo Park. Robust regression for image binarization under heavy noise and nonuniform background. *Pattern Recognition*, 81:224–239, Elsevier, 2018.

[157] Quang Nhat Vo, Soo Hyung Kim, Hyung Jeong Yang, and Gueesang Lee. Binarization of degraded document images based on hierarchical deep supervised network. *Pattern Recognition*, 74:568–586, Elsevier, 2018.

[158] Florian Westphal, Håkan Grahn, and Niklas Lavesson. Efficient document image binarization using heterogeneous computing and parameter tuning. *International Journal on Document Analysis and Recognition*, 21(1-2):41–58, Springer, 2018.

[159] Florian Westphal, Niklas Lavesson, and Håkan Grahn. Document image binarization using recurrent neural networks. In *Document Analysis Systems (DAS)*, pages 263–268. IEEE, 2018.

[160] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* The MIT Press, 1964.

[161] C. Wolf, J.-M. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. In *International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1037–1040, 2002.

[162] Yue Wu, Premkumar Natarajan, Stephen Rawls, and Wael AbdAlmageed. Learning document image binarization from data. In *International Conference on Image Processing (ICIP)*, pages 3763–3767. IEEE, 2016.

[163] Marcel Würsch, Marcus Liwicki, and Rolf Ingold. Web services in document image analysis - recent developments on divaservices and the importance of building an ecosystem. In *Document Analysis Systems (DAS)*, pages 334–339. IEEE, 2018.

[164] Wei Xiong, Jingjing Xu, Zijie Xiong, Juan Wang, and Min Liu. Degraded historical document image binarization using local features and support vector machine (SVM). *Optik*, 164:218–223, Elsevier, 2018.

[165] E. Zemouri, Youcef Chibani, and Youcef Brik. Restoration based contourlet transform for historical document image binarization. In *International Conference on Multimedia Computing and Systems (ICMCS)*, pages 309–313. IEEE, 2014.

Chapter 3

Document Image Binarization with Fully Convolutional Neural Networks

Abstract

Binarization of degraded historical manuscript images is an important pre-processing step for many document processing tasks. We formulate binarization as a pixel classification learning task and apply a novel Fully Convolutional Network (FCN) architecture that operates at multiple image scales, including full resolution. The FCN is trained to optimize a continuous version of the Pseudo F-measure metric and an ensemble of FCNs outperform the competition winners on 4 of 7 DIBCO competitions. This same binarization technique can also be applied to different domains such as Palm Leaf Manuscripts with good performance. We analyze the performance of the proposed model w.r.t. the architectural hyperparameters, size and diversity of training data, and the input features chosen.

Tensmeyer, Chris, and Tony Martinez. “Document image binarization with fully convolutional neural networks.” *International Conference on Document Analysis and Recognition (ICDAR).* Vol. 1, pp. 99-104. IEEE, 2017.

3.1 Introduction

To minimize the impact of physical document degradation on document image analysis tasks (e.g. page segmentation, OCR), it is often desirable to first binarize the digital images. Such degradations include non-uniform background, stains, faded ink, ink bleeding through the page, and un-even illumination. Binarization separates the raw content of the document from these noise factors by labeling each pixel as either foreground ink or background. This is a well-studied problem, even for highly degraded documents, as evidenced by the popularity of the Document Image Binarization Content (DIBCO) competitions [6, 12, 17–21].

We present a learning model for document image binarization. Fully Convolutional Networks (FCN) [10] alternate convolution and non-linear operations to efficiently classify all pixels of an input image in a single forward pass. While FCNs have been previously proposed for semantic segmentation tasks, their output predictions are poorly localized due to high downsampling. We propose a multi-scale architectural variation that results in precise localization and preserves the generalization benefits of downsampling.

FCNs can be applied to diverse domains of documents images without tuning hyper-parameters. For example, we show that the same FCN architecture can be trained to achieve state-of-the-art performance on both historical paper documents (i.e. DIBCO data) and on palm leaf manuscripts. Current state-of-the-art methods for paper documents (e.g.[8]) tend to perform poorly on palm leaf manuscripts due to domain differences between the tasks [3]. FCNs also automatically adapt to any particular definition of binarization ground truth (e.g. see [23]) because they do not incorporate explicit prior information.

Many common binarization approachs compute local or global thresholds based on image statistics [13, 22]. One disadvantage of this approach is the threshold is invariant to a permutation of the pixels, i.e., statistics ignore shape. On the other hand, other approaches (e.g. edge detection, Markov Random Field (MRF), connected components) include strong biases about the shape of foreground components. In contrast, FCNs learn from training data to exploit the spatial arrangements of pixels without relying on a hand-crafted bias on local shapes.

Previous learning approaches are trained to optimize per-pixel accuracy, which is problematic due to the majority of pixels being background. These methods typically resort to heuristic sampling of pixels to achieve a balanced training set. Instead, we propose directly optimizing a continuous adaptation of the Pseudo F-measure (P-FM) [11] which has been a DIBCO evaluation metric since 2013 [20]. Because P-FM does not penalize foreground border pixels that are predicted as background, we combine P-FM with regular F-measure (FM) so the FCN correctly classifies border pixels.

The contributions of this work are as follows. First, we propose the use of FCNs for document image binarization and determine good architectures for this task. We show that directly optimizing the proposed continuous Pseudo F-measure exceeds the previous state-of-the-art on DIBCO competition data. We compute a learning curve and show that diversity of data is more important than quantity of data. Finally, we demonstrate that FCN performance can be improved though additional input features, including the output of other binarization algorithms.

3.2 Related Work

Howe's method formulates image binarization as energy minimization over a MRF [8]. The unary energy terms are computed from the image Laplacian and the pairwise connections are determined by Canny Edge detection [4]. An exact minimization of the energy function can be obtained by solving the equivalent Max Flow problem [2]. Variants of this method have placed 1$^{\text{st}}$ in HDIBCO 2012, 2014, 2016 and 2$^{\text{nd}}$ in DIBCO 2013 [12, 19–21]. The DIBCO 2013 winner combines local image contrast and local image gradients to determine edges between text and background [24].

Several classification-based binarization approaches have been proposed. Hamza et al. train a Multi-Layer Perceptron (MLP) classifier on pixel labels derived from clustering [7].

Kefali et al. train an MLP using ground truth data to classify each pixel based on the intensity values of its 3x3 neighborhood and the global image mean and standard deviation [9], though they do not out perform Sauvola's method [22]. Afzal et al. trained a 2D Long Short Term Memory (LSTM) network which incorporates both local and global information [1]. Their approach greatly reduced OCR error on the binarized text compared to Sauvola.

Pastor-Pellicer et al. explored the use of Convolutional Neural Networks (CNN) to classify each pixel given its 19x19 neighborhood of intesity values [15]. They report an FM of 87.74 on DIBCO 2013 compared to 92.70 achieved by the competition winner. Wu et al. trained an Extremely Randomized Trees classifier using a wide variety of statistical and heuristic features extracted from various neighborhoods around the pixel of interest. Because the number of background pixels greatly exceeds the number of foreground pixels, they heuristically sampled a training set to balance both classes. In contrast, we directly optimize the Pseudo F-measure instead of determining the precision/recall tradeoff through sampling.

Long et al. proposed FCNs for the more general semantic segmentation problem in natural images [10]. Both Zheng et al. and Chen et al. combined Conditional Random Fields (CRF) with FCNS to improve localization and consistency of predictions [5, 26]. These FCNs heavily downsample inputs, which results in poor localization. Thus prior FCNs are not good models for document image binarization.

3.3 Methods

In this section, we describe FCNs, our multi-scale extension, and the proposed loss function we use to train them.

3.3.1 Fully Convolutional Networks

Fully Convolutional Networks (FCN) are models composed of alternating convolution operations with element-wise non-linearities. The FCNs we consider map an input image

$x \in \mathbb{R}^{D \times H \times W} \to y \in \mathbb{R}^{H \times W}$, where $y_{ij} \in [0, 1]$ is the probability that pixel x_{ij} is foreground, and D is the number of input channels (e.g. 3 for RGB).

Specifically, the ℓ^{th} layer $(1 \leq \ell \leq L)$ of the FCN performs a convolution operation with learnable kernels followed by a rectification:

$$x_\ell = \text{ReLU}(W_\ell * x_{\ell-1} + b_\ell) \tag{3.1}$$

where $x_\ell \in \mathbb{R}^{D_\ell \times H \times W}$ is the output of layer ℓ, x_0 is the input image, $\text{ReLU}(z) = \max(0, z)$ is element-wise rectification, $W_\ell \in \mathbb{R}^{D_\ell \times K_\ell \times K_\ell \times D_{\ell-1}}$ are convolution kernels, and $b_\ell \in \mathbb{R}^{D_\ell}$ is a bias term for each kernel. To obtain probabilities for each pixel, a sigmoid function is applied element-wise to $x_L \in \mathbb{R}^{1 \times H \times W}$ without rectification.

Each W_ℓ can be interpreted as D_ℓ distinct convolution kernels (i.e. $W_{\ell,i} \in \mathbb{R}^{K_\ell \times K_\ell \times D_{\ell-1}}$) that have two spatial dimensions, each of size K_ℓ, and a depth dimension of size $D_{\ell-1}$, which is the number of kernels used in the previous layer. Thus convolution is carried out in two spatial dimensions, but each $W_{\ell,i}$ spans all channels of its layer's input. Each of these kernels is applied independently to $x_{\ell-1}$ to yield a single channel $x_{\ell,i} \in \mathbb{R}^{H \times W}$. Each $x_{\ell,i}$ then becomes a single channel of the multi-channel x_ℓ.

In our paper, we simplify the choice of hyper-parameters by using the same value for many parameters, regardless of ℓ. For example, we primarily use $D_\ell = 64$ and $K_\ell = 9$.

3.3.2 Multi-Scale

Many applications of FCNs to dense pixel prediction problems find that incorporating information at multiple scales leads to significantly improved performance. This is an elegant way to fuse local and global features for classification. For example, the original FCN [10] downsamples images to $\frac{1}{32}$ the input size and finds improved performance by incorporating features computed at scales $\frac{1}{8}$ and $\frac{1}{16}$. In contrast, binarizing text images requires precise localization which is difficult when images are downsampled.

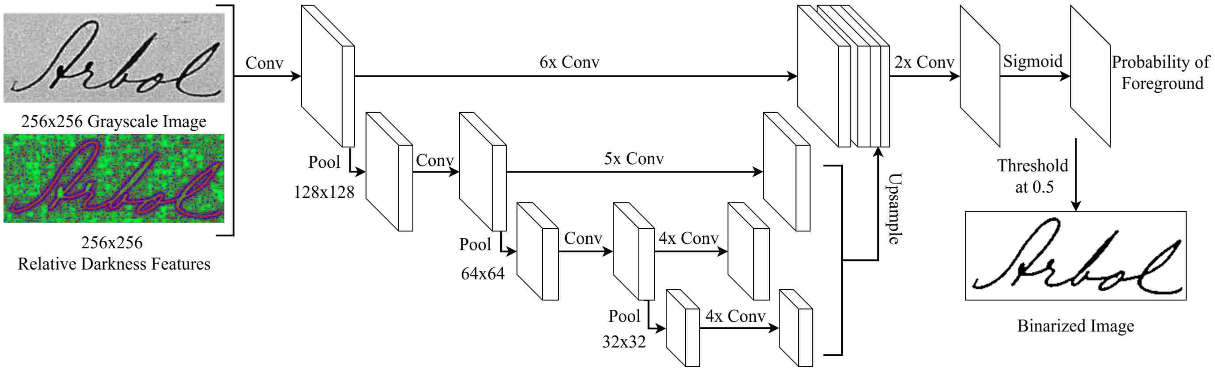


Figure 3.1: Multi-Scale Fully Convolutional Neural Network architecture. All Conv operations are followed by ReLU, and "Nx Conv" labels indicate N successive conv layers. All pool layers downsample by a factor of 2. Upsampling is done by bilinear interpolation to the input spatial dimensions.

We utilize a branching FCN to compute features over scales $\frac{1}{1}$, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$. This is shown in Figure 3.1. This network has depth $L = 9$, width $D_\ell = 64$, 4 scales, and kernel size $K_\ell = 9$. At certain layers, 2x2 average pooling is applied to produce an additional branch of the FCN at a smaller scale. After several convolution layers at each scale, the output of each scale is upsampled to the original size using bilinear interpolation. This is followed by two more convolution layers applied to the concatenated scale outputs. This allows the FCN to fuse both local and increasingly global features to classify pixels.

3.3.3 Pseudo F-measure Loss

Pseudo F-measure (P-FM) was formulated with the intuition that binarization errors on textual images should be penalized based on how they might obscure individual characters [11]. For example, false positives (actual background predicted as foreground) far from foreground components have a smaller penalty than false positives between characters. P-FM is the harmonic mean of Pseudo Recall and Pseudo Precision, which we denote R_{ps} and P_{ps} respectively. These quantities are computed as

$$F_{ps} = \frac{2R_{ps}P_{ps}}{R_{ps} + P_{ps}} \tag{3.2}$$

$$R_{ps} = \frac{\Sigma_{xy}(B_{xy}G_{xy}R^W_{xy})}{\Sigma_{xy}(G_{xy}R^W_{xy})} \tag{3.3}$$

$$P_{ps} = \frac{\Sigma_{xy}(G_{xy}B_{xy}P^W_{xy})}{\Sigma_{xy}(B_{xy}P^W_{xy})} \tag{3.4}$$

where B_{xy} is predicted probability of foreground for the pixel (x, y), $G_{xy} \in \{0, 1\}$ is the ground truth class of pixel (x, y), and R^W and P^W are per-pixel weights for recall and precision errors respectively.

While R^W and P^W can be arbitrarily specified, we compute them using [11] to yield the P-FM used in the DIBCO evaluation protocol. Using uniform weights yields the more common F-measure, which was explored in [16] as the training objective of neural networks.

To use Eq. 3.2 as the objective function in Stochastic Gradient Descent (SGD) training of the FCN, we must derive an expression for $\frac{dF}{dB}$. For simplicity, we drop the subscripts on F_{ps}, R_{ps}, P_{ps}. Taking the derivative of Eq. 3.2, we have

$$\frac{dF}{dB} = \frac{\partial F}{\partial P}\frac{dP}{dB} + \frac{\partial F}{\partial R}\frac{dR}{dB} = 2\frac{\frac{dR}{dB}P^2 + \frac{dP}{dB}R^2}{(P+R)^2} \tag{3.5}$$

Noting that $\frac{dR}{dB}$ is a matrix of partial derivatives, deriving $\frac{\partial R}{\partial B_{ij}}$ is simple as B only occurs in the numerator in Eq. 3.3.

$$\frac{\partial R}{\partial B_{ij}} = \frac{\partial}{\partial B_{ij}}\left[\frac{\Sigma_{xy}(B_{xy}G_{xy}R^W_{xy})}{\Sigma_{xy}(G_{xy}R^W_{xy})}\right] = \frac{G_{ij}R^W_{ij}}{\Sigma_{xy}(G_{xy}R^W_{xy})} \tag{3.6}$$

Similarly, we can derive $\frac{\partial P}{\partial B_{ij}}$ by applying the quotient rule for derivatives to Eq. 3.4 and simplifying.

$$\frac{\partial P}{\partial B_{ij}} = P^W_{ij}\frac{\Sigma_{xy}(B_{xy}P^W_{xy})G_{ij} - \Sigma_{xy}(G_{xy}B_{xy}P^W_{xy})}{[\Sigma_{xy}(B_{xy}P^W_{xy})]^2} \tag{3.7}$$

3.3.4 Datasets and Metrics

We primarily use two datasets for evaluation: DIBCOs [6, 12, 17–21] and Palm Leaf Manuscripts (PLM) [3]. For DIBCO experiments, we used all 86 competition images from the years 2009-2016 for either training, validation, or test data. When testing on a particular DIBCO year, that year's competition data composes the test set, 10 random other images compose the validation set, and the remaining images compose the training set. For PLM, we randomly split the 50 training images into 40 training and 10 validation and use the designated 50 images for testing [3]. While there are two sets of ground truth annotations for PLM, for simplicity we use only the first set.

To avoid overfitting of testing data, development of our method was performed using the HDIBCO 2016 validation set. The test data was only used in the final evaluation and not used to select models, architectures, or features.

The metrics we report are from the DIBCO 2013 evaluation criteria: P-FM, FM, Peak Signal to Noise Ratio (PSNR), and Distance Reciprocal Distortion (DRD). For P-FM, FM, and PSNR, higher numbers indicate better performance. For DRD, lower is better.

3.3.5 Implementation Details

We used the deep learning library Caffe for all experiments. Our training and validation sets are composed of 256x256 crops extracted at a stride of 64 pixels from the input images. Image crops composed of only background pixels are discarded as Pseudo Recall is undefined for all background predictions. Our FCNs take as input the gray scale image plus locally computed Relative Darkness features (see Section 3.4.5).

To binarize a whole image, the FCN binarizes individual overlapping 256x256 crops and extracts the center 128x128 patch of each output crop (except for border regions). This ensures each pixel has sufficient context for the FCN to classify it. These patches are stitched together to form the whole binarized image.

Dataset	Loss	Metrics			
		P-FM	FM	DRD	PSNR
HDIBCO 2016	P-FM	**94.09 (94.67)**	86.66 (87.06)	4.62 (4.38)	17.73 (17.86)
	FM	92.90 (93.23)	89.93 (90.30)	3.69 (3.51)	**18.73 (18.90)**
	P-FM + FM	93.22 (93.76)	89.01 (89.52)	4.01 (3.76)	18.48 (18.67)
	Cross-Entropy	92.59 (92.94)	**90.20 (90.56)**	**3.62 (3.45)**	18.68 (18.84)
PLM	P-FM	68.23 (68.55)	66.93 (67.20)	9.24 (9.10)	14.79 (14.83)
	FM	67.40 (67.74)	**68.38 (68.69)**	9.86 (9.68)	14.59 (14.64)
	P-FM + FM	**68.54 (68.96)**	68.27 (68.63)	**9.12 (8.94)**	**14.81 (14.87)**
	Cross-Entropy	66.41 (66.77)	65.38 (65.68)	9.95 (9.78)	14.58 (14.63)

Table 3.1: Average performance of 5 FCNs on H-DIBCO 2016 and PLM datasets for various loss functions. Numbers in parenthesis indicate ensemble performance.

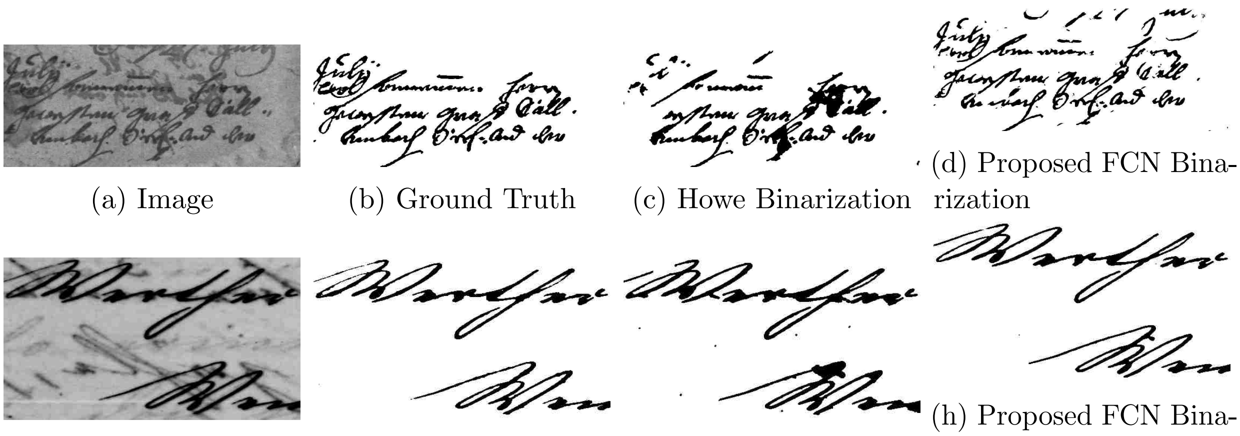
We used SGD to train FCNs with an initial learning rate (LR) of 0.001, mini-batch size of 10 patches, and an L2 weight decay of 0.0005. Gradient clipping to an L2 norm of 10 was employed to help stablize learning [14]. The LR was multiplied by 0.1 if performance on the validation set failed to improve for 1.5 epochs. Training ended when the LR decayed to 10^{-6} and the set of parameters that best performed on the validation set is the output of the training procedure. During training, we stochastically add a small constant to each channel of the input image as color jitter, which empirically improves performance by a small amount.

3.4 Experiments

In this section, we give results and discussion for our experiments to validate the proposed model.

3.4.1 Loss Functions

In this set of experiments, we compare 4 loss functions on HDIBCO 2016 and PLM: P-FM, FM, P-FM + FM, and Cross Entropy (CE). Because the P-FM loss function does not penalize the outer border of foreground components being classified as background (due to recall weights of 0), we experimented with adding together P-FM and FM losses during training. CE loss is a standard classification based loss that optimizes per-pixel accuracy.



(a) Image (b) Ground Truth (c) Howe Binarization (d) Proposed FCN Binarization

(e) Image (f) Ground Truth (g) Howe Binarization (h) Proposed FCN Binarization

Figure 3.2: Qualitative comparison of proposed ensemble of FCNs with state-of-the-art Howe Binarizataion [8]. Images contain significant bleed through noise and come from the H-DIBCO 2016 test data.

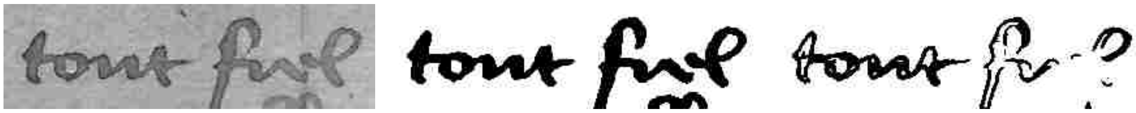
For each loss function, we trained 5 FCNs from different random initializations. Table 3.1 shows both the average performance of individual FCNs as well as the combined ensemble performance in parenthesis. The outputs of individual FCNs are combined using per-pixel majority vote. This improves performance by a significant amount at the cost of increased computation.

Unsurprisingly, optimizing for P-FM (or P-FM + FM) yields the best performance for P-FM, with CE performing worst for P-FM. Training with only P-FM lowers performance in other metrics due to predicting border pixels as background. While developing our method, FM loss generally out-performed CE loss on all metrics on validation data, though surprisingly CE performed better on the HDIBCO 2016 test data. Nevertheless, based on our validation set performance, we chose to use P-FM + FM loss for the remaining experiments. For reference, the best published FM on PLM is 68.76 [3], which was achieved by a single scale FCN that was pretrained on DIBCO and proprietary data.

Figure 3.2 shows a qualitative comparison between Howe's method [8] (using the author's code) and the ensemble of 5 FCNs trained with P-FM + FM loss. Because Howe's method uses pixel connectivity, it sometimes misclassifies small background regions as foreground when they are bordered by both foreground ink and bleed-through noise. On the

	Proposed Method				Best Competition System			
Competition	P-FM	FM	DRD	PSNR	P-FM	FM	DRD	PSNR
DIBCO 2009	92.59	89.76	4.89	18.43	-	**91.24**	-	**18.66**
H-DIBCO 2010	97.65	**94.89**	1.26	**21.84**	-	91.50	-	19.78
DIBCO 2011	97.70	**93.60**	**1.85**	**20.11**	-	88.74	5.36	17.97
H-DIBCO 2012	96.67	92.53	**2.48**	20.60	-	**92.85**	2.66	**21.80**
DIBCO 2013	**96.81**	**93.17**	**2.21**	20.71	94.19	92.70	3.10	**21.29**
H-DIBCO 2014	94.78	91.96	2.72	20.76	**97.65**	**96.88**	**0.90**	**22.66**
H-DIBCO 2016	**93.76**	**89.52**	**3.76**	**18.67**	91.84	88.72	3.86	18.45

Table 3.2: Comparison of ensemble of 5 FCNs with DIBCO competition winners. The "-" symbols indicate unreported metrics.



(a) Image (b) Grond Truth (c) Proposed Binarization

Figure 3.3: Failure case on H-DIBCO 2014. None of the training images are similar, so FCN generalizes poorly to this kind of ink.

other hand, FCNs compute local features on multiple scales, so it learns to recognize whether text is bleed-through noise or actual foreground ink.

3.4.2 DIBCO Performance

Here we compare our ensemble of 5 FCNs with the best performing competition submissions for each of the 7 DIBCO competitions. The winning systems did not necessarily have the best performance wrt all metrics. Therefore, we compare with the best score per metric for any system entered in the competition. For example, in DIBCO 2013, the rank 1 system had the highest P-FM (94.19), but the rank 2 system had the highest FM (92.70).

The results are presented in Table 3.2. For 4 competitions, including the most recent HDIBCO 2016, the ensemble of FCNs out-performs the best submitted entries. Our FM is low on DIBCO 2009 due to false positives located far away from the foreground ink. Under the P-FM, these mistakes count for less because they don't obscure the readability of the text. For HDIBCO 2014, our method performs very poorly on image 6 (see Figure 3.3), where it

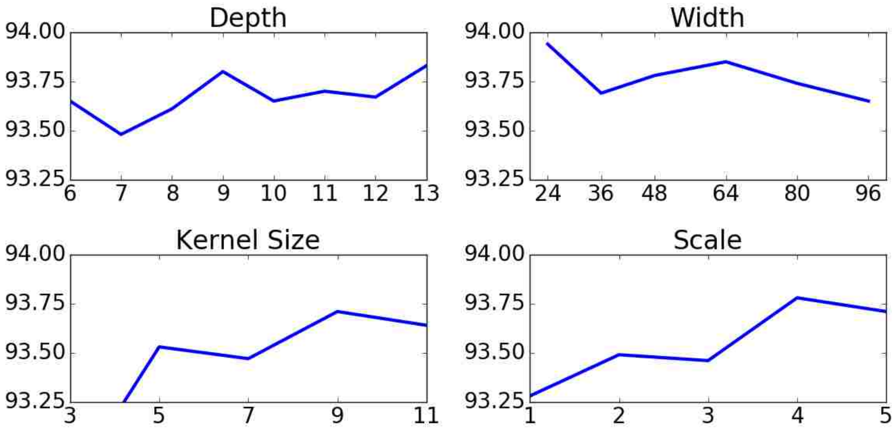


Figure 3.4: P-FM on HDIBCO 2016 as a function of architecture hyperparameters evaluated with an ensemble of 5 FCNs. In general, performance is not sensitive to any particular combination of hyperparameters.

achieves a P-FM of 56.4. On the other 9 H-DIBCO 2014 images, it achieves almost perfect performance with an average P-FM of 99.04.

3.4.3 Architecture Search

FCNs have a number of hyperparameters that together constitute the architecture of the network. Here we independently vary four architecture components and measure P-FM on the HDIBCO 2016 dataset. Specifically, we vary network depth, width, number of scales, and kernel size. The base architecture (used in previous experiments) is depth $L = 9$, width $D_\ell = 64$, 4 scales, and kernel size $K_\ell = 9$.

The results for ensembles of 5 FCN for each architecture are presented in Figure 3.4. Performance seems relatively insensitive to architectural hyperparameters, with the exception that kernel sizes of 3 perform worse than larger kernels. Improvement due to ensemble prediction is typically larger than performance differences among architectures.

3.4.4 How Much Data Is Enough?

More data typically leads to better performing learning models, though eventually adding more data yields diminishing returns. Here we analyze the amount of data need by constructing a

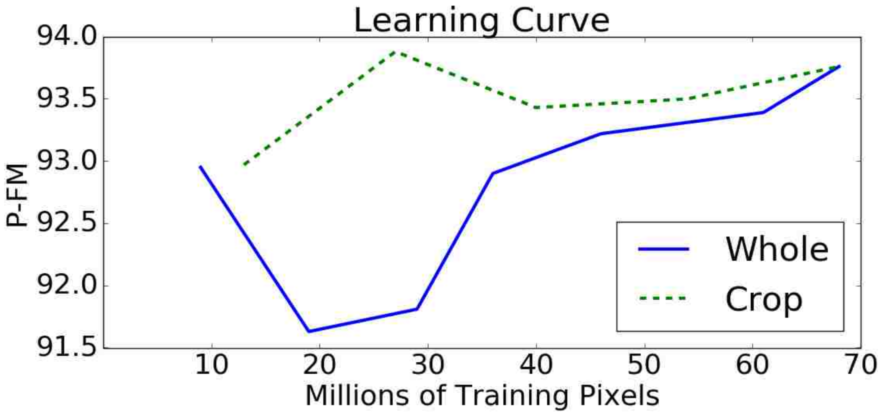


Figure 3.5: Learning curve for two methods of shrinking the training data. The solid blue line indicates that whole images were removed from the training set. The dashed green line indicates that all training images were kept, but were cropped to be smaller.

learning curve with amount of data on the x-axis and P-FM on the y-axis for the HDIBCO 2016 dataset. Figure 3.5 shows the curve for two different ways of varying the amount of training data (keeping the validation data the same). The first way simply reduces the number of training images along the x-axis. This decreases the diversity of images that compose the training set. The second way retains all 76 training/validation images, but crops each image to be 20%, 40%, 60%, or 80% of its original width. This also decreases the number of pixels for training, but retains diversity of inputs presented to the FCN.

Even at similar number of pixels, cropping training images instead of removing them performs better and indicates that greater diversity in the training data is a key factor for improving performance. This is logical because if an image is homogeneous in handwriting style and noise content, then many of the pixels are locally similar. Thus, if the goal of manual or semi-automatic creation of ground truth binarizations is to create more training data, the annotator should focus on small, diverse images instead of fully annotating large images.

Training on 10M pixels resulted in better test set performance than on 20-40M pixels for whole images. We find this as evidence of overfitting the validation set, which is used for model selection. The validation P-FM for 10M and 20M pixels are 92.84 and 95.26

Extra Feature	Size	P-FM	Extra Feature	Size	P-FM
None	-	96.63	10th Percentile	3	96.69
Min	9	96.73	25th Percentile	3	96.63
Max	3	96.74	Canny Edges	-	96.93
Median	39	96.57	Percentile Filter	-	96.16
Mean	39	96.38	Bilateral Filter	100	96.01
Otsu	-	95.55	Standard Deviation	3	95.65
Howe	-	97.14	Relative Darkness	5	**97.15**

Table 3.3: P-FM on HDIBCO 2016 by including additional input features. The "-" symbols indicate globally computed features.

respectively. Additionally, the added training images are more similar to the 10 validation images than the 10 test images. This shows that there is significant dataset bias in evaluation with so few number of images, even if the number of pixels is very large.

3.4.5 Input Features

While FCNs can learn good discriminative features from raw pixel intensities, there may exist useful features that the FCN is not able to learn because they cannot be efficiently approximated by alternating convolution and rectification operations. For example, Sauvola's method uses local standard deviation which is not easy for an FCN to learn.

We experimented with input features that are densely computed and treated as additional input image channels. For this experiment, we used a batch size of 5 and minimized P-FM loss. We report P-FM on HDIBCO 2016 validation set for an ensemble of 5 FCNs in Table 3.3 for the best performing parameterization of each feature type.

Relative Darkness (RD) [25] features performed best, though using the output of Howe's method [8] as an input feature performs almost as well. RD is computed by counting the number of pixels in a local window that are darker, lighter, and similar to the central pixel. We used RD features with a window size of 5x5 and a similarity threshold of ±10 in all experiments in this paper.

3.5 Conclusion

In this work, we have proposed FCNs trained with a combined P-FM and FM loss for the task of document image binarization. Our proposed method handles diverse domains of documents. It out performs the competition winners for 4 of 7 DIBCO competitions and is competitive with the state-of-the-art on Palm Leaf Manuscripts. We analyzed the architecture and found that performance is stable wrt changes in the architecture. We found that the number of training images (i.e. diverse training data) is a more important factor than number of pixels used in training. Finally, we analyzed using additional features as input to the FCN and found that Relative Darkness features [25] and the output of Howe binarization [8] perform best.

References

[1] Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki. Deepdocclassifier: Document classification with deep convolutional neural network. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE, 2015.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, IEEE, 2004.

[3] Jean-Christophe Burie, Mickaël Coustaty, Setiawan Hadi, Made Windu Antara Kesiman, Jean-Marc Ogier, Erick Paulus, Kimheng Sok, I. Made Gede Sunarya, and Dona Valy. ICFHR 2016 competition on the analysis of handwritten text in images of balinese palm leaf manuscripts. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 596–601, 2016.

[4] John Canny. A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, IEEE, 1986.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv preprint arXiv:1412.7062*, 2014.

[6] Basilios Gatos, Konstantinos Ntirogiannis, and Ioannis Pratikakis. ICDAR 2009 document image binarization contest (DIBCO 2009). In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 9, pages 1375–1382. IEEE, 2009.

[7] Hatem Hamza, Eddie Smigiel, and E. Belaid. Neural based binarization techniques. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 317–321. IEEE, 2005.

[8] Nicholas R. Howe. Document binarization with automatic parameter tuning. *International Journal on Document Analysis and Recognition*, 16(3):247–258, Springer, 2013.

[9] Abderrahmane Kefali, Toufik Sari, and Halima Bahi. Foreground-background separation by feed-forward neural networks in old manuscripts. *Informatica*, 38(4), 2014.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 3431–3440. IEEE, 2015.

[11] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. Performance evaluation methodology for historical document image binarization. *Transactions on Image Processing*, 22(2):595–609, IEEE, 2013.

[12] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. ICFHR2014 competition on handwritten document image binarization (H-DIBCO 2014). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 809–813. IEEE, 2014.

[13] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, IEEE, 1979.

[14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (ICML)*, 28: 1310–1318, 2013.

[15] J. Pastor-Pellicer, S. España-Boquera, F. Zamora-Martínez, M. Zeshan Afzal, and Maria Jose Castro-Bleda. Insights on the use of convolutional neural networks for

document image binarization. In *International Work-Conference on Artificial Neural Networks*, pages 115–126. Springer, 2015.

[16] Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España-Boquera, and María José Castro-Bleda. F-measure as the error function to train neural networks. In *International Work-Conference on Artificial Neural Networks*, pages 376–384. Springer, 2013.

[17] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. H-DIBCO 2010 - handwritten document image binarization competition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 727–732. IEEE, 2010.

[18] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2011 document image binarization contest (DIBCO 2011). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1506–1510. IEEE, 2011.

[19] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. ICFHR 2012 competition on handwritten document image binarization (H-DIBCO 2012). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 817–822. IEEE, 2012.

[20] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2013 document image binarization contest (DIBCO 2013). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1471–1476. IEEE, 2013.

[21] Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. ICFHR2016 handwritten document image binarization contest (H-DIBCO 2016). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 619–623. IEEE, 2016.

[22] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, Elsevier, 2000.

[23] Elisa H. Barney Smith and Chang An. Effect of "ground truth" on image binarization. In *Document Analysis Systems (DAS)*, pages 250–254. IEEE, 2012.

[24] Bolan Su, Shijian Lu, and Chew Lim Tan. Robust document image binarization technique for degraded document images. *Transactions on Image Processing*, 22(4):1408–1417, IEEE, 2013.

[25] Yue Wu, Premkumar Natarajan, Stephen Rawls, and Wael AbdAlmageed. Learning document image binarization from data. In *International Conference on Image Processing (ICIP)*, pages 3763–3767. IEEE, 2016.

[26] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, pages 1529–1537. IEEE, 2015.

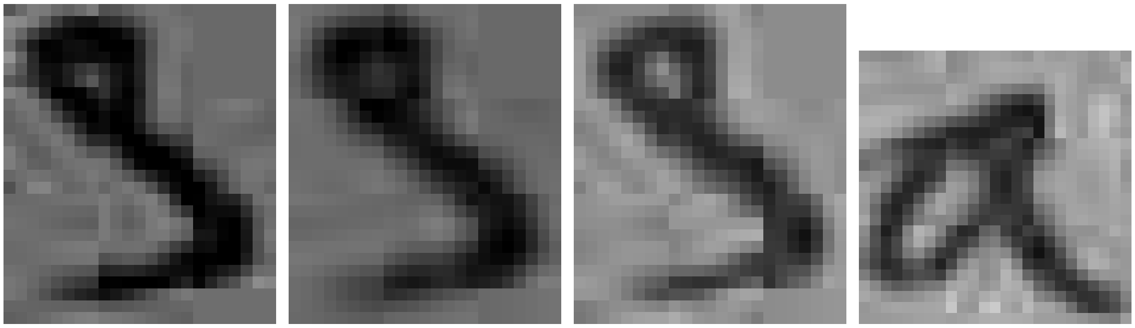
Chapter 4

Generating Realistic Binarization Data with Generative Adversarial Networks

Abstract

One of the limitations for using Deep Learning models to solve binarization tasks is that there is a lack of large quantities of labeled data available to train such models. Efforts to create synthetic data for binarization mostly rely on heuristic image processing techniques and generally lack realism. In this work, we propose a method to produce realistic synthetic data using an adversarially trained image translation model. We extend the popular CycleGAN model to be conditioned on the ground truth binarization mask as it translates images from the domain of synthetic images to the domain of real images. For evaluation, we train deep networks on synthetic datasets produced in different ways and measure their performance on the DIBCO datasets. Compared to not pretraining, we reduce error by 13% on average, and compared to pretraining on unrealistic data, we reduce error by 6%. Visually, we show that DGT-CycleGAN model produces more realistic synthetic data than other models.

Submitted to *International Conference on Document Analysis and Recognition (ICDAR)*. Chris Tensmeyer, Mike Brodie, Daniel Saunders, and Tony Martinez. "Generating Realistic Binarization Data with Generative Adversarial Networks."



(a) Composite (b) Blurred (c) Proposed (d) Real

Figure 4.1: Synthetic data created by compositing clean foreground on noisy background (a) does not appear realistic, even after blurring the edges (b). Instead, we train a neural network to learn a mapping from (a) to (c), under the criteria that the refined data is indistinguishable from real data (d).

4.1 Introduction

Recently, deep learning models have achieved state-of-the-art performance on binarization of historical documents that contain handwritten and machine printed text [3, 35, 36], largely surpassing heuristic approaches. However, the performance of deep models is tied to the quality, amount, and variety of the data used to train the model [35]. There are few manually labeled images available to train (and evaluate) models for binarization tasks, so the performance of deep models is limited by the amount of training data.

Synthetic data is a cheap way to obtain more labeled data, but previous heuristic methods for data generation lack realism at the pixel level (see Figure 4.1). Typically, clean foreground, in order to have the ground truth (GT), is combined with noisy background with added blurring [14], ink fading [12], lighting artifacts [22], etc. However, it is difficult to explictly model the large variety of degradations that occur in real data. Additionally, no comparative study has evaluated different ways of generating synthetic data in the context of training deep learning models for binarization.

In this paper, we propose a novel method to generate realistic synthetic data for binarization tasks with pixel GT. Because explicit modeling of document degradations proves challenging, our method uses a black-box adversarially trained deep neural network to refine heuristically generated synthetic data into images that appear more realistic. To do this, we extended CycleGAN [37], which is an unpaired image-to-image translation model that learns to convert images from one domain to another. While the basic CycleGAN model produces good data, we find that modifying the discriminator to condition on the binarization GT leads to increased realism and better agreement between the GT and the produced image. We call our proposed model DGT-CycleGAN.

We validate our approach by pretraining deep networks on synthetic datasets generated by DGT-CycleGAN, CycleGAN, and image compositing. We then evaluate both pretrained only and finetuned models on each of the DIBCO datasets [8, 20, 23–29]. Pretraining on DGT-CycleGAN produced data leads to the best performance of all the pretrained models. Finetuning on DIBCO data after pretraining DGT-CycleGAN data leads to 13% F-measure error reduction compared to no pretraining and to 6% error reduction compared to pretraining on non-realistic synthetic data.

4.2 Related Works

Here we review related work on synthetic data used for binarization and deep learning models proposed to perform binarization.

4.2.1 Synthetic Data for Binarization

The lack of labeled data for binarization has not gone unnoticed in the Document Image Analysis (DIA) community, and several approaches utilizing synthetic data have been proposed. An ICFHR 2010 binarization competition evaluated entrants on synthetic data produced by simple compositing of clean PDF foreground on real noisy backgrounds [21, 34]. In [14], compositing is used to combine scanned foreground images with synthetic backgrounds produced by modeling the background textures of real historical documents. While these backgrounds are uniform and free of stains, they do add artificial bleed through with controllable strength by compositing blurred text. Such images have been used in the context of method evaluation, but not for model training.

The open source DocCreator library [12] contains several tools for creating noisy synthetic data including paper deformation, uneven lighting, adaptive blur, bleed through, and ink degradation. Synthetic images produced by DocCreator have been shown to improve a performance prediction model for binarization algorithms. DIVA Document Image Degradation (DID) [33] is a complementary technique that uses gradient domain editing to realistically insert stains and spots to an existing image.

4.2.2 Deep Models for Binarization

Several deep learning models have been proposed in recent years and typically outperform other methods in binarization. For example, in the 2017 Document Image Binarization Competition (DIBCO) [28], the 4 top performing groups all used deep models.

Encoder-decoder architectures have been popular for binarization [3, 4, 17, 22]. In [22], the model is trained entirely with synthetic data that are created from clean images by introducing blur, salt-pepper noise, and lighting artifacts. Meng et al. used novel regularization terms to encourage label smoothness and encode apriori knowledge about foreground region intensity [17].

The multi-scale Fully Convolutional Network (FCN) architecture used in [35] showed that variety of training data was a more important factor than number of pixels. The PDNet model [2] is comprised of an FCN that makes per-pixel predictions and a Primal-Dual network that enforces label smoothness. Both networks were pretrained end-to-end on synthetic data, and then finetuned on real data. Vo et al. proposed fusing the output of 3 Deeply Supervised Networks (DSN) that are independently trained at multiple scales and achieved excellent results.

4.3 Generating Realistic Synthetic Data

The goal of our method is to generate both realistic synthetic data for binarization and corresponding so-called “ground truth” labeling that can be used for training and evaluation of binarization methods. We first generate synthetic images by compositing clean foreground text on noisy backgrounds. The corresponding GT is obtained by applying a global threshold to the clean foreground image to produce a binary mask. To make the data realistic, we then refine the synthetic images using a model that has learned to translate images from the synthetic domain to the domain of real images. Translating images from synthetic to real domains can be accomplished by a model such as CycleGAN [37]; however, we find that modifying the CycleGAN model to condition on the GT of the synthetic image leads to better agreement between the modified synthetic images and the GT.

For clarity, we will use the term *refined* to refer to images that have been modified to look realistic by a model (Section 4.3.2) and reserve the term *synthetic* to only refer to images that are produced by compositing (Section 4.3.1).

4.3.1 Compositing

Before refining the synthetic data to look realistic, we generate the synthetic data by compositing clean foreground with noisy background. To obtain the foreground ink, we use the IAM database of handwritten images [16]. This dataset was created by having various writers copy a prompt on a clean white piece of paper, so our foreground text has a wide variety of character shapes, stroke widths, etc.

To create the foreground for a new image, we randomly choose a page from the IAM dataset. We then start near the upper left corner of the page and randomly sample a sequence of words, stitching them together with random line and word spacing until the page region is filled. This image is easily binarized with a global threshold to obtain the GT for this sample.

We then use alpha blending to composite the foreground text onto a noisy background region taken from a real degraded document. For 30% of images, we insert artificial bleed-

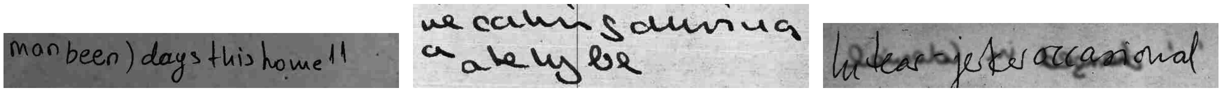


Figure 4.2: Example synthetic images produced by compositing clean foreground on noisy background. The rightmost sample includes artificial bleed-through noise.

through by producing a reversed text image that is heavily blurred and then alpha blended onto the synthetic image. Some example text lines from images produced in this manner are shown in Figure 4.2.

4.3.2 Refining Synthetic Data

After generating composited synthetic data and the corresponding GT, the data is refined using a deep model to look more realistic. Our proposed method modifies the CycleGAN model [37] to be conditioned on the GT so that the refined images are consistent with their GT. Before explaining our modification, we give an overview of the CycleGAN model.

CycleGAN is composed of 4 models: 2 image-to-image *generators* and 2 image-to-probability *discriminators*:

- $G_{A \to B}$ converts images from domain A to domain B
- $G_{B \to A}$ converts images from domain B to domain A
- D_A predicts probability of belonging to domain A
- D_B predicts probability of belonging to domain B

In this work, domain A is a set of synthetic images (Section 4.3.1). Domain B is the set of historical documents that we are interested in binarizing, which could be, e.g. DIBCO-like images or Palm Leaf Manuscripts.

The 4 models are jointly trained using two training sets of unpaired images (no labels required). By unpaired, we mean that there is no correspondence between the images in the training sets for A and B. We denote $a_i \in A$ to be an image from domain A with similar notation for B.

Name	Model Updated	Loss
Cycle-consistent	$G_{A \to B}, G_{B \to A}$	$G_{B \to A}(G_{A \to B}(a_i)) \approx a_i$
Identity	$G_{A \to B}$	$G_{A \to B}(b_i) \approx b_i$
Recognize real	D_B	$D_B(b_i) \approx 1$
Reject fake	D_B	$D_B(G_{A \to B}(a_i)) \approx 0$
Fool D	$G_{A \to B}$	$D_B(G_{A \to B}(a_i)) \approx 1$

Table 4.1: Losses for CycleGAN model. The second column indicates which models update their weights for that loss.

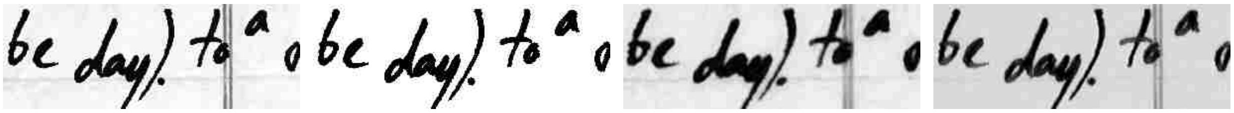
Ten losses are jointly optimized, based on the 5 criteria shown in Table 4.1. The other 5 losses are obtained by switching A for B, e.g. $G_{A \to B}(G_{B \to A}(b_i)) \approx b_i$ for the reversed cycle consistent loss. The last two criteria in Table 3.1 are contradictory and form the basis for adversarial learning, where two models optimize opposing criteria. In theory, with a single pair of models, $G_{A \to B}$ and D_B can reach an equilibrium where $G_{A \to B}$ perfectly models domain B and D_B cannot distinguish the generated images from real images [9]. In this paired scenario, $G_{A \to B}$ and $G_{B \to A}$ not only learn to model their respective domains by competing with D_A and D_B, but also learn to preserve the coarse structure of their input images in order to satisfy the cycle-consistent and identity losses.

After training the 4 models, we throw away 3 and keep only $G_{A \to B}$, which has learned to transform synthetic images to refined images that appear more realistic. The generators are fully convolutional and produce an output image the same size as its input image.

4.3.3 Ground Truth Consistency

The refined images produced by the basic CycleGAN model are problematic because the character boundaries are often subtly changed. This means that the refined image is inconsistent with the GT produced by thresholding the original clean foreground text. Figure 4.3 shows an example.

We solve this problem by modifying the discriminator for the real domain, D_B to take as input both an image and its corresponding GT. This is done by concatenating the



(a) Composite (b) Ground Truth (c) CycleGAN (d) DGT-CycleGAN

Figure 4.3: Naive use of CycleGAN to refine composited synthetic data (a) can lead to mismatch between the refined image (c) and the ground truth (b). This is solved by conditioning the discriminator of CycleGAN on the ground truth (d).

grayscale image with the GT mask to produce a single input for D_B. We call this model DGT-CycleGAN.

While all of the refined images produced by $G_{A \to B}$ have corresponding GT by construction, not all $b_i \in B$ have GT. Surprisingly, we found that for b_i that do not have GT, setting the GT mask to all background was sufficient to enable learning. We also tried inputting to D_B only the b_i with GT, but the refined images suffered in quality.

We also experimented with other CycleGAN variants to improve GT consistency including conditioning $G_{A \to B}$ on the GT and cross domain regularization, i.e. a loss to encourage $G_{A \to B}(a_i) \approx a_i$. While these two varaints produced visually better GT agreement compared to CycleGAN, the visual agreement was not as good as that of DGT-CycleGAN, nor was their quantiative results as compelling as DGT-CycleGAN (data not shown).

4.4 Experiments

To validate our proposed method, we generated a synthetic dataset using compositing (Section 4.3.1). We then generated different refined versions of this synthetic data using both CycleGAN (Section 4.3.2) and our proposed DGT-CycleGAN (Section 4.3.3). To quantitatively measure the realism of each of these datasets, we trained deep neural networks on each dataset and measured their binarization performance on DIBCO datasets from 2009-2018. We then finetuned each network on DIBCO data, and measured their performance again.

While binarization performance is not a direct measure of data realism or of consistency between images and their GT, it is one possible use case for generating synthetic data, so it makes sense to evaluate our method in this manner. We now give the implementation details of the experiments.

4.4.1 Data

The foreground ink used to produce the synthetic data comes from the IAM dataset of handwritten images [16]. The background images were produced by cropping background regions from real historical documents. We obtained these regions from a variety of sources including commerical websites[1], Google image search, gallica collections[2], and public datasets including Saintgall [6], Parzival [7], RVL-CDIP [10], Rodrigo [32], CBAD [5], Historical Layout Analysis Competitions [1], and Bentham [31]. From 1700 background regions we extracted 1000 images.

We also collected images to represent the domain of real documents. For this dataset, we collected 181 labeled images from DIBCO 2009-2018 [8, 20, 23–29], PHIBD [18], and the Irish Script on Screen (ISOS) bleed-through dataset [30]. We also collected 210 unlabeled documents from the same sources as the background regions and produced 468 cropped images of text regions.

The above mentioned synthetic and real data are used to produce the refined datasets. To evaluate the realism of these datasets, we evaluate the performance of models trained on the synthetic and refined datasets using the labeled DIBCO data, and report the standard DIBCO metrics including F-measure, Pseudo F-measure [19], Peak Signal-to-Noise Ratio (PSNR), and Distance Reciprocal Distortion [15].

Models finetuned on labeled DIBCO data are done in a leave-one-dataset-out fashion. For example, when reporting results for a finetuned model on DIBCO 2009, the model was finetuned and validated using DIBCO data from 2010-2018. Additionally, for DGT-CycleGAN

[1] `ancestry.com` and `familysearch.org`

[2] `https://gallica.bnf.fr/`

results, we created a refined dataset for each DIBCO year by omitting that year's GT data when training the DGT-CycleGAN model.

4.4.2 Model Details

To produce the refined data, we modified the official CycleGAN code[3] and used many of the default parameters for network architecture and training. We use the default architecture for D_A and D_B, which is a 5-layer PatchGAN network [11] that classifies patches of the input image as real or fake. We experimented with two generator architectures, including the default 24-layer residual encoder-decoder architecture that downsamples by a factor of 4. The other generator architecture was a 20-layer residual architecture that removed the downsampling and upsampling blocks from the default model. For DGT-CycleGAN, we used the 20-layer generator architecture.

For training, we used the ADAM optimizer [13] for 100 epochs with a batch size of 32 and a default learning rate of 0.0002. To construct a batch, we sampled a random 256x256 crop from each of the sampled images.

The model used for binarization training is taken from [35]. For each synthetic or refined dataset, we pretrained 5 models. We then finetuned 5 models for each DIBCO dataset, each one starting with distinct pretrained weights. As in [35], the output of the 5 models is combined by averaging probabilities to produce a final output.

Models were pretrained for 40,000 iterations with a batch size of 10 256x256 patches. Model selection was performed using a random validation set taken from the same synthetic or refined dataset used for training. Finetuning on DIBCO data continued until performance stopped improving on a random validation set of DIBCO data.

Method	2009	2010	2011	2012	2013	2014	2016	2017	2018	Avg
Composite-pretrain	13.54	13.28	42.21	15.86	46.58	72.49	5.00	12.84	14.92	26.30
CycleGAN-G24-pretrain	81.95	88.40	81.35	85.77	82.06	91.46	81.10	81.56	76.53	83.35
CycleGAN-G20-pretrain	81.92	88.97	83.37	**87.67**	85.43	89.70	**82.68**	**82.14**	80.27	84.68
DGT-CycleGAN-pretrain	**84.76**	**89.68**	**84.50**	84.95	**87.63**	**93.52**	81.05	80.92	**82.46**	**85.50**
No-pretrain	90.62	93.69	89.97	90.85	93.43	93.63	90.20	90.43	80.02	90.32
Composite-finetune	91.45	94.44	91.81	91.22	94.57	95.88	89.86	90.88	80.88	91.22
CycleGAN-G24-finetune	91.78	94.95	92.25	**93.09**	94.83	96.29	90.44	**91.11**	79.61	91.59
CycleGAN-G20-finetune	**92.00**	**95.01**	**92.63**	93.06	**94.99**	96.44	90.08	90.88	80.53	**91.74**
DGT-CycleGAN-finetune	91.52	94.45	91.57	92.27	94.87	**96.69**	**90.45**	90.94	**81.78**	91.62
DIBCO winners [8, 20, 23–29]	91.24	91.50	88.74	92.85	92.70	96.88	88.72	91.04	88.34	91.33
FCN ensemble [35]	89.76	94.89	93.60	92.53	93.17	91.96	89.52	-	-	-
DSN ensemble [36]	-	-	93.3	-	94.4	96.61	90.10	-	-	-

Table 4.2: Detailed F-measure results on all DIBCO datasets

Method	F-measure	Recall	Precision	Pseudo F-measure	Pseudo Recall	Pseudo Precision	DRD	PSNR
Composite-pretrain	26.30	20.01	87.64	29.90	23.62	87.42	17.44	12.44
CycleGAN-G24-pretrain	83.35	77.05	93.19	90.24	89.40	92.47	5.97	16.93
CycleGAN-G20-pretrain	84.68	78.83	92.95	**91.82**	**92.28**	92.07	5.37	16.68
DGT-CycleGAN-pretrain	**85.50**	**79.92**	**94.27**	90.87	89.81	**93.42**	**5.12**	**17.55**
No-pretrain	90.32	87.67	94.00	94.22	95.92	93.30	3.68	19.21
Composite-finetune	91.22	89.35	93.96	94.45	96.34	**93.96**	3.45	19.72
CycleGAN-G24-finetune	91.59	90.27	93.67	94.67	97.10	92.97	3.37	19.94
CycleGAN-G20-finetune	**91.74**	**90.37**	93.80	**94.85**	**97.26**	93.07	3.28	**19.98**
DGT-CycleGAN-finetune	91.62	89.77	**94.20**	94.84	96.75	93.51	**3.22**	19.91

Table 4.3: Standard DIBCO metrics averaged over all DIBCO datasets

4.5 Results

Our quantitative results are presented in Tables 4.2 and 4.3. Table 4.2 presents the F-measure on each DIBCO dataset for the models pretrained on different datasets both with and without finetuning. The top rows are models that are only pretrained on synthetic or refined data, and the middle set of rows are the corresponding models after finetuning on DIBCO data. The baseline method that involves no pretraining and only training on DIBCO data is include in this group. The bottom group of rows shows some reported numbers from the literature, including the performance of the top system submitted in each DIBCO competition. However, it is difficult to make direct comparisons because of differences in training data.

Table 4.3 shows several more metrics for the same models, but only averaged over all the DIBCO datasets. This gives a richer characterization of the predictions as the various metrics capture different facets of performance.

[3] `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`

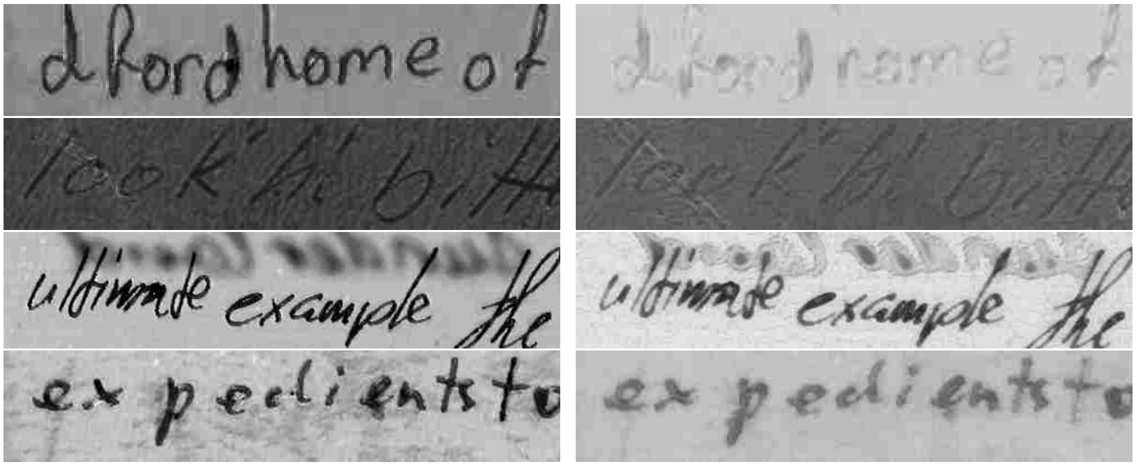


Figure 4.4: Refined data produced by the proposed DGT-CycleGAN (left column) compared with CycleGAN-G20 (right column)

4.5.1 Pretraining Results

Pretraining using the refined data produced by DGT-CycleGAN yields the highest F-measure in 6 of 9 DIBCO datasets (Table 4.2) and the best average performance in 6 out of 8 metrics (Table 4.3). This suggests that the refined data produced by DGT-CycleGAN is more realistic and/or has better agreement with the generated GT compared the data produced by the basic CycleGAN models.

We also see that CycleGAN-G20 outperforms CycleGAN-G24 on pretraining, indicating that downsampling inside the generator model leads to poorer performance. As expected, the naively created composited dataset performs extremely poorly as pretraining data due to the clear separation between foreground and background.

Figure 4.4 shows some example images produced by DGT-CycleGAN and CycleGAN-G20. Visual inspection of the model prediction errors indicate that DGT-CycleGAN is better at classifying bleed-through noise as background, while CycleGAN-G20 does better at recovering thin strokes in handwritten text (but not machine printed). This makes sense as CycleGAN-G20 tends to refine images by making them uniformly lighter, so much of the data it produces has light strokes. However, this makes it more challenging to classify bleed-through text, which is typically light, as background. In contrast, DGT-CycleGAN produces darker text that has appropriate texture and realistic local changes in ink intensity.

4.5.2 Finetuning Results

The finetuning results do not indicate a clear best method for data generation because the differences among the three CycleGAN based models in F-measure, Pseudo F-measure, PSNR and DRD are small (< 0.2). However, CycleGAN-G20 always outperforms CycleGAN-G24 and outperforms DGT-CycleGAN on 5 out of 8 metrics. The realism of the pretraining dataset likely matters less for finetuning because issues like GT consistency can be corrected by finetuning on data with quality GT.

Overall, we see improvement in many metrics by pretraining on synthetic and refined data. Training on refined data produced by DGT-CycleGAN reduces F-measure error by 13% compared to not pretraining and 6% compared to pretraining on synthetic data. Even pretraining on the unrealistic composited synthetic data causes performance after finetuning to increase significantly (9% error reduction).

4.6 Conclusion

In this work, we have proposed a new way to generate synthetic data with pixel ground truth for binarization tasks. Instead of explicitly modeling the large variety of degradations that occur in real historical documents, we rely on black box modeling with our proposed DGT-CycleGAN model to transform unrealistic synthetic data into refined images that appear realistic. We have shown that pretraining deep neural networks on the more realistic synthetic data leads to better predictive performance both before and after finetuning on real data.

One direction for future work is experimenting with better generation of the initial synthetic data (e.g. [12, 33]) and seeing how this affects the quality of the refined data produced by DGT-CycleGAN. Another direction is examining if the realistic synthetic data generated in this work is useful in the evaluation of binarization algorithms and how it compares to the ground truth produced in semi-automated ways.

References

[1] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, and Stefan Pletschacher. ICDAR 2013 competition on historical newspaper layout analysis (HNLA 2013). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1454–1458. IEEE, 2013.

[2] Kalyan Ram Ayyalasomayajula, Filip Malmberg, and Anders Brun. PDNet: Semantic segmentation integrated with a primal-dual network for document binarization. *Pattern Recognition Letters*, pages 52–60, Elsevier, 2018.

[3] Jorge Calvo-Zaragoza and Antonio-Javier Gallego. A selectional auto-encoder approach for document image binarization. *Pattern Recognition*, 86:37–47, Elsevier, 2019.

[4] Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. Pixel-wise binarization of musical documents with convolutional neural networks. In *International Conference on Machine Vision Applications (MVA)*, pages 362–365. IEEE, 2017.

[5] Markus Diem, Florian Kleber, Stefan Fiel, Tobias Grüning, and Basilis Gatos. CBAD: ICDAR2017 competition on baseline detection. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1355–1360. IEEE, 2017.

[6] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. Transcription alignment of Latin manuscripts using hidden markov models. In *International Workshop on Historical Document Imaging and Processing (HIP)*, pages 29–36. ACM, 2011.

[7] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7): 934–942, Elsevier, 2012.

[8] Basilios Gatos, Konstantinos Ntirogiannis, and Ioannis Pratikakis. ICDAR 2009 document image binarization contest (DIBCO 2009). In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 9, pages 1375–1382. IEEE, 2009.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680. MIT Press, 2014.

[10] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE, 2017.

[12] Nicholas Journet, Muriel Visani, Boris Mansencal, Kieu Van-Cuong, and Antoine Billy. DocCreator: A new software for creating synthetic ground-truthed document images. *Journal of Imaging*, 3(4):62, Multidisciplinary Digital Publishing Institute, 2017.

[13] Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *International Conference on Learning Representation (ICLR)*, ArXiv, 2014.

[14] Rafael Dueire Lins, Marcos Martins de Almeida, Rodrigo Barros Bernardino, Darlisson Jesus, and José Mário Oliveira. Assessing binarization techniques for document images. In *Symposium on Document Engineering (DocEng)*, pages 183–192. ACM, 2017.

[15] Haiping Lu, Alex C. Kot, and Yun Q. Shi. Distance-reciprocal distortion measure for binary document images. *Signal Processing Letters*, 11(2):228–231, IEEE, 2004.

[16] U. V. Marti and Horst Bunke. A full English sentence database for off-line handwriting recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, page 705. IEEE, 1999.

[17] Gaofeng Meng, Kun Yuan, Ying Wu, Shiming Xiang, and Chunhong Pan. Deep networks for degraded document image binarization through pyramid reconstruction. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 727–732. IEEE, 2017.

[18] Hossein Ziaei Nafchi, Seyed Morteza Ayatollahi, Reza Farrahi Moghaddam, and Mohamed Cheriet. An efficient ground truthing tool for binarization of historical manuscripts. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 807–811. IEEE, 2013.

[19] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. Performance evaluation methodology for historical document image binarization. *Transactions on Image Processing*, 22(2):595–609, IEEE, 2013.

[20] Konstantinos Ntirogiannis, Basilis Gatos, and Ioannis Pratikakis. ICFHR2014 competition on handwritten document image binarization (H-DIBCO 2014). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 809–813. IEEE, 2014.

[21] Roberto Paredes, Ergina Kavallieratou, and Rafael Dueire Lins. ICFHR 2010 contest: Quantitative evaluation of binarization algorithms. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 733–736. IEEE, 2010.

[22] Xujun Peng, Huaigu Cao, and Prem Natarajan. Using convolutional encoder-decoder for document image binarization. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 708–713. IEEE, 2017.

[23] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. H-DIBCO 2010 - handwritten document image binarization competition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 727–732. IEEE, 2010.

[24] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2011 document image binarization contest (DIBCO 2011). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1506–1510. IEEE, 2011.

[25] Ioannis Pratikakis, Basilis Gatos, and Konstantinos Ntirogiannis. ICFHR 2012 competition on handwritten document image binarization (H-DIBCO 2012). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 817–822. IEEE, 2012.

[26] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis. ICDAR 2013 document image binarization contest (DIBCO 2013). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1471–1476. IEEE, 2013.

[27] Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. ICFHR2016 handwritten document image binarization contest (H-DIBCO 2016). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 619–623. IEEE, 2016.

[28] Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. ICDAR2017 competition on document image binarization (DIBCO 2017). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1395–1403. IEEE, 2017.

[29] Ioannis Pratikakis, Konstantinos Zagoris, Panagiotis Kaddas, and Basilis Gatos. ICFHR 2018 competition on handwritten document image binarization contest (H-DIBCO 2018).

In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 1–1. IEEE, 2018.

[30] Róisín Rowley-Brooke, François Pitié, and Anil Kokaram. A ground truth bleed-through document image database. In *International Conference on Theory and Practice of Digital Libraries*, pages 185–196. Springer, 2012.

[31] Joan Andreu Sánchez, Verónica Romero, Alejandro H. Toselli, and Enrique Vidal. ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 785–790. IEEE, 2014.

[32] Nicolás Serrano, Francisco Castro, and Alfons Juan. The RODRIGO database. In *Conference on International Language Resources and Evaluation (LREC)*, pages 19–21. European Languages Resources Association (ELRA), 2010.

[33] Mathias Seuret, Kai Chen, Nicole Eichenbergery, Marcus Liwicki, and Rolf Ingold. Gradient-domain degradations for improving historical documents images layout analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1006–1010. IEEE, 2015.

[34] Pyrrhos Stathis, Ergina Kavallieratou, and Nikos Papamarkos. An evaluation technique for binarization algorithms. *Journal of Universal Computer Science*, 14(18):3011–3030, 2008.

[35] Chris Tensmeyer and Tony Martinez. Document image binarization with fully convolutional neural networks. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 99–104. IEEE, 2017.

[36] Quang Nhat Vo, Soo Hyung Kim, Hyung Jeong Yang, and Gueesang Lee. Binarization of degraded document images based on hierarchical deep supervised network. *Pattern Recognition*, 74:568–586, Elsevier, 2018.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

Part II

Document Image Segmentation

Chapters 5-7 treat the topic of document segmentation, where the document image is divided in its logical parts. Chapters 5 and 6 cover the basic task of identifying the boundaries of a document page from the rest of the image so that later document processing algorithms are only applied to the page region. Chapter 5 takes a pixel-labeling approach to identifying the page region, while Chapter 6 presents a general keypoint detection model that can be used to identify the corners of a document. Chapter 7 presents a pair of deep models that partition an image or PDF of a table into cells, rows, and columns to recover the structure of the table.

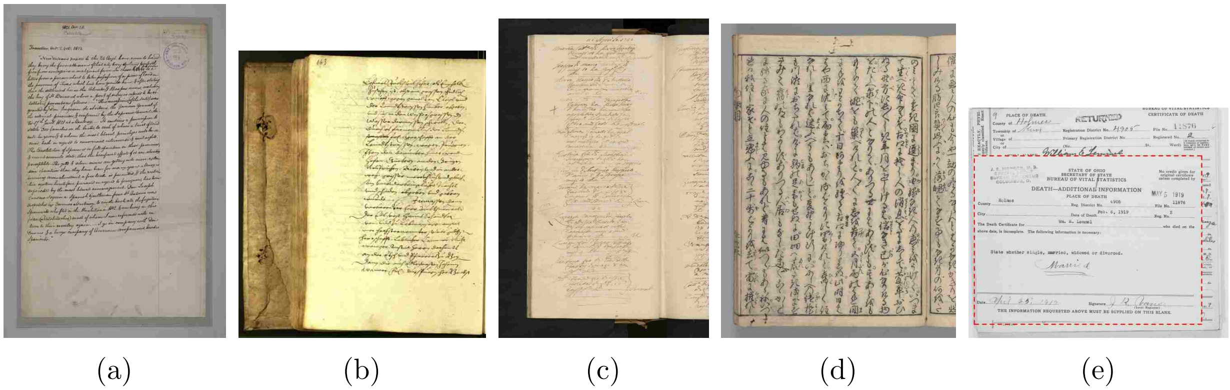
Chapter 5

PageNet: Page Boundary Extraction in Historical Handwritten Documents

Abstract

When digitizing a document into an image, it is common to include a surrounding border region to visually indicate that the entire document is present in the image. However, this border should be removed prior to automated processing. In this work, we present a deep learning system, PageNet, which identifies the main page region in an image in order to segment content from both textual and non-textual border noise. In PageNet, a Fully Convolutional Network obtains a pixel-wise segmentation which is post-processed into a quadrilateral region. We evaluate PageNet on 4 collections of historical handwritten documents and obtain over 94% mean intersection over union on all datasets and approach human performance on 2 collections. Additionally, we show that PageNet can segment documents that are overlayed on top of other documents.

Tensmeyer, Chris, Brian Davis, Curtis Wigington, Iain Lee, and Bill Barrett. “PageNet: Page Boundary Extraction in Historical Handwritten Documents.” *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing (HIP).* pp. 59-64. ACM, 2017.



(a) (b) (c) (d) (e)

Figure 5.1: Examples of common border noise. (a-d) Background around the border. (b-d) Book edges. (c-d) Textual noise due to neighboring pages. (e) Textual noise due to one document (red square) overlayed on top of another.

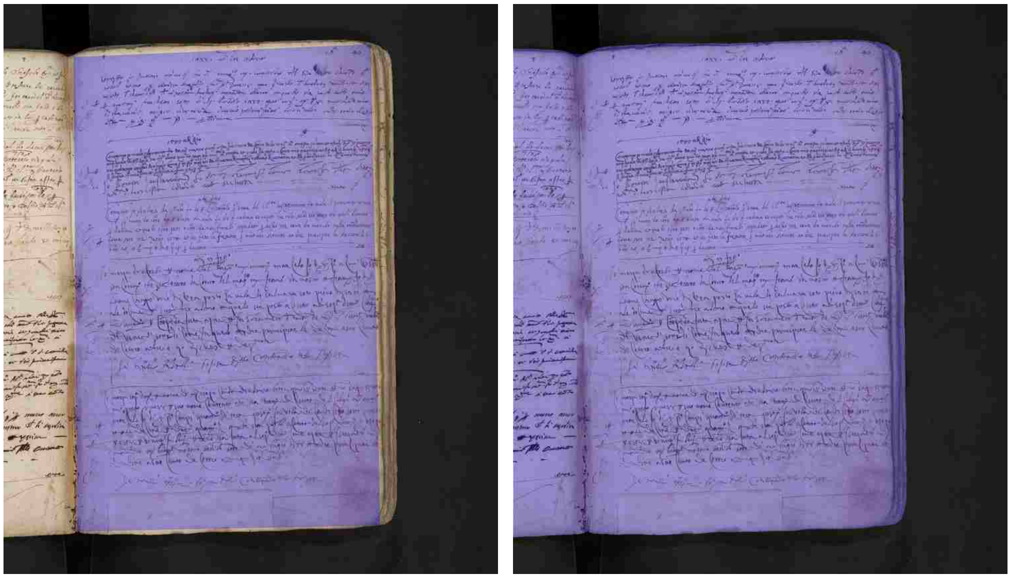
5.1 Introduction

When digitizing a document into an image, it is common to include a surrounding border region to visually indicate that the entire document is present in the image. The *border noise* resulting from this process can interfere with document analysis algorithms and cause undesirable results. For example, text may be detected and transcribed outside of the main page region when textual noise is present due to a partially visible neighboring page. Thus, it is beneficial to preprocess the image to remove the border region.

Some examples of border noise in historical documents include background, book edges, overlayed documents, and parts of neighboring pages (see Figure 5.1). While removing background is feasible using simple segmentation techniques, other types of border noise are more challenging.

A number of approaches have been developed to remove border noise (e.g., [3, 5, 7, 19, 21]). However, as noted in [4], many prior methods make assumptions that do not necessarily hold for historical handwritten documents. These assumptions about the input image include consistent text size, absolute location of border noise, straight text lines, and distances between page text and border [4].

In this work, we propose *PageNet*, a deep learning based system that, given an input image, predicts a bounding quadrilateral for the *main page region* (see Figure 5.2). PageNet is



(a) PageNet Segmentation (b) GrabCut Segmentation

Figure 5.2: PageNet, our proposed system, segments the main page region from border noise such as book edges and portions of the opposite page. Traditional segmentation algorithms like GrabCut struggle with some types of border noise, though are effective at removing the background.

composed of a Fully Convolutional Network (FCN) and post-processing. The FCN component outputs a pixel-wise segmentation which is post-processed to extract a quadrilateral shaped region. Detecting the main page region is similar to finding the *page frame* [19], but the former task detects the entire page while the latter crops the detected region to the page text. PageNet is able to robustly handle a variety of border noise because, as a learning based system, it does not make any explicit assumptions about the border noise, layout, or content of the input image. Though learning methods can potentially overfit the training collection, we demonstrate that PageNet generalizes to other collections of documents.

We experimentally evaluated PageNet on 5 collections of historical documents that we manually annotated with quadrilateral regions. To estimate human agreement on this task, we made a second set of annotations for a subset of images. On our primary test-set, PageNet achieves 97.4% mean Intersection over Union (mIoU) while the human agreement is 98.3% mIoU. On all collections tested, we achieve a mIoU of >94%. Additionally, we show that PageNet is capable of segmenting documents that are overlayed on top of other documents.

In order to support the reproducibility of our work, we are releasing our code, models, and dataset annotations, available at `https://github.com/ctensmeyer/pagenet`.

5.2 Related Work

We take a segmentation approach to removing border noise, so we review the literature on traditional border noise removal techniques and on segmentation.

Fan et al. remove non-textual marginal scanning noise (e.g., large black regions) from printed documents by detecting noise with a resolution reduction approach and removing noise through region growing or local thresholding [7]. Shafait et al. handle both textual and non-textual marginal noise by finding the page frame of an image that maximizes a quality function w.r.t. an input layout analysis composed of sets of connected components, text lines, and zones [20]. The method of Shafait and Breuel examines the local densities of black and white pixels in fixed image regions to identify noise and also removes connected components near the image border [19]. Stamatopoulos et al. proposed a system based on projection profiles to find the two individual page frames in images of books where two pages are shown. They report an average F-measure of 99.2% on 15 historical books [21]. Bukhari et al. find the page frame for camera captured documents by detecting text lines, aligning the text line end points, and estimating a straight line from the endpoints using RANdom SAmple Consensus (RANSAC) linear regression [3]. For further reading, we refer the reader to a recent survey on border noise removal [4].

Our approach differs from the previously mentioned border noise removal techniques because we do not make assumptions about the location of the document in the image. Furthermore, our task differs from the page frame based methods, which find a tight bounding box around the page content. In contrast, we attempt to detect the entire main page region, including background regions. Our approaches are complimentary as once the main page region is detected, the page frame is more easily obtained.

Another formulation of the border noise removal problem is to find the four corners of the bounding quadrilateral of the page, which is a sub-task shared with perspective dewarping techniques. Jagannathan et al. find page corners in camera captured documents by identifying two sets of parallel lines and two sets of perpendicular lines in the perspective transformed image [12]. Yang et al. use a Hough line transform to detect boundaries in binarized images of license plates [24].

Intelligent scissors segments an object from background by finding a least cost path through a weighted graph defined over pixels, subject to input constraints [17]. Active Contours or Snakes formulate segmentation as a continuous optimization problem and finds object boundaries by minimizing a boundary energy cost and the cost of deformation from some prior shape [14]. Graph Cut methods (e.g.[1]) formulate image segmentation as finding the minimum cut over a graph constructed from the image. Weights in the graph are determined by pixel colors and by per-pixel apriori costs of being assigned to the foreground and background segments. GrabCut iteratively performs graph cut segmentations, starting from an initial rough bounding box. The result of each iteration is used to refine a color model which is used to construct the edge weights for the next iteration [18].

Several neural network approaches have been proposed for image segmentation. The Fully Convolution Network (FCN) learns an end-to-end classification function for each pixel in the image [16]. However, the output is poorly localized due to downsampling in the FCN architecture used. Zheng et al. integrated a Conditional Random Field (CRF) graphical model into the FCN to improve segmentation localization [25]. In contrast, our approach maintains the input resolution and therefore does not suffer from poor localization. The Spatial Transformer Network [11] learns a latent affine transformation in conjunction with a task specific objective, effectively learning cropping, rotation, and skew correct in an end-to-end fashion. In our case, we are interested in directly learning a pre-processing transformation from ground truth. Chen and Seuret used a convolutional network to classify super pixels as background, text, decoration, and comments [6]. Super pixel based features would not work in our case as neighboring pages are identical to the main page region in local appearance.

5.3 Method

In this section, we describe PageNet, which takes in a document image and outputs the coordinates of four corners of the quadrilateral that encloses the main page region. PageNet has two parts:

1. A Fully Convolutional Neural Network (FCN) to classify pixels as page or background
2. Post processing to extract a quadrilateral region

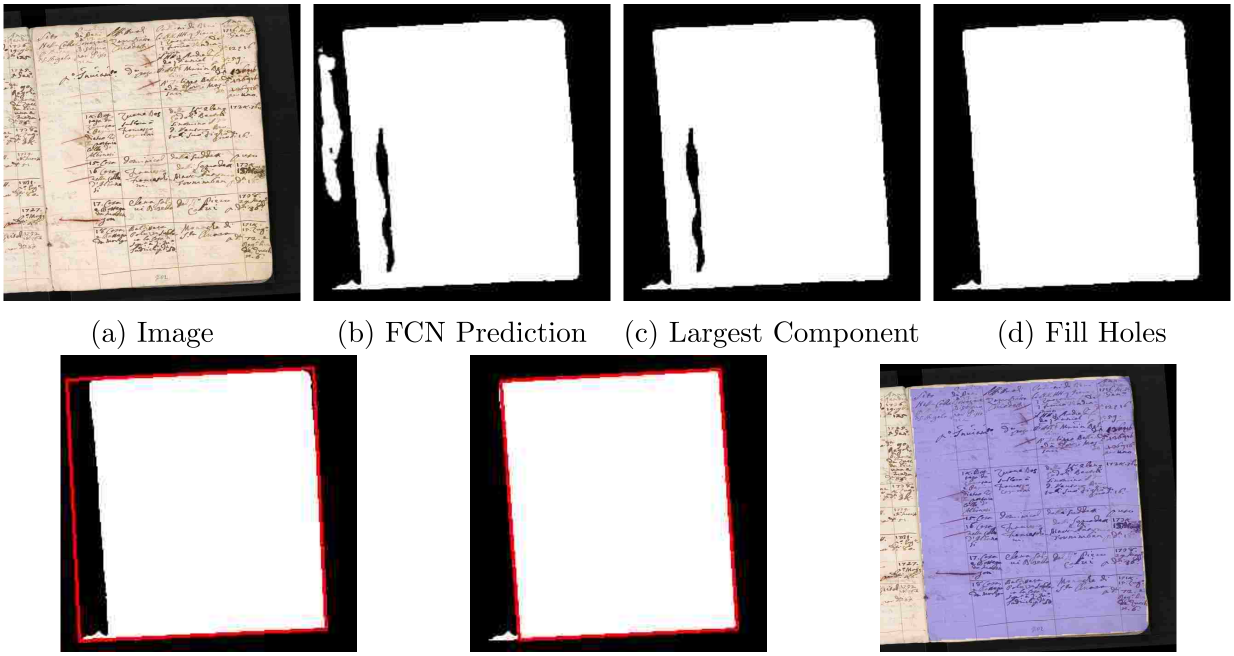
5.3.1 Pixel Classification

FCNs are learning models that alternate convolutions with element-wise non-linear functions [16]. They differ from traditional CNNs (e.g., AlexNet) that have fully connected layers which constrain the input image size. In particular, the FCN used in PageNet maps an input RGB image $x \in \mathbb{R}^{3 \times H \times W} \to y \in \mathbb{R}^{H \times W}$, where $y_{ij} \in [0, 1]$ is the probability that pixel x_{ij} is part of the main page region.

Each layer of a basic FCN performs the operation

$$x_\ell = g(W_\ell * x_{\ell-1} + b_\ell) \tag{5.1}$$

where ℓ is a layer index, $*$ indicates multi-channel 2D convolution, W_ℓ is a set of learnable convolution filters, b_ℓ is a learnable bias term, and g is a non-linear function. In our case, we use $g(z) = \text{ReLU}(z) = \max(0, z)$ as element-wise non-linear rectification. In some FCN architectures, spatial resolution is decreased at certain layers through pooling or strided convolution and increased through bilinear interpolation or backwards convolution [16]. In the last layer of the network, ReLU is replaced with a channel-wise softmax operation (over 2 channels in our case) to obtain output probabilities for each pixel.



(a) Image (b) FCN Prediction (c) Largest Component (d) Fill Holes

(e) Oriented Bounding Rect. (f) Quad. Maximizing IoU (g) Result

Figure 5.3: Post processing to extract quadrilateral from FCN predictions.

PageNet uses a successful multi-scale FCN architecture originally designed for binarizing handwritten text [22]. This FCN operates on 4 image scales: 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$. The full resolution image scale uses 7 sequential layers (Eq. 5.1), and each smaller layer uses one layer less than the previous (e.g. $\frac{1}{8}$ scale uses 4 layers). The input feature maps of the 3 smallest scales are obtained by 2×2 average-pooling over the output of the first layer of the next highest image scale. The FCN concatenates the output feature maps of each scale, upsampling them to the input image size using bilinear interpolation. Thus the architecture both preserves the original input resolution and benefits from a larger context window obtained through downsampling. This is followed by 2 more convolution layers and the softmax operation. For full details on the FCN architecture, we refer the reader to [22]. We performed initial experiments (not shown) with a single scale FCN but it performed worse, likely due to smaller surrounding context for each pixel.

5.3.2 Quadrilaterals

Thresholding output of the FCN yields a binary image (Figure 5.3b), which is converted to the coordinates of a quadrilateral around the main page region in the image. While the pixel representation may be already useful for some applications (e.g., masking text detection regions), it can lack global and local spatial consistency due to the FCN predicting pixels independently based on local context. Representing the detected page region as a quadrilateral fixes some errors in the FCN output and makes it easier for potential downstream processing to incorporate this information.

Figure 5.3 demonstrates the following post-processing steps that converts the binary per-pixel output of the FCN (after thresholding at 0.5 probability) to a quadrilateral region:

1. Remove all but the largest foreground components.
2. Fill in any holes in the remaining foreground component.
3. Find the minimum area oriented bounding rectangle using Rotating Calipers method [23].
4. Iteratively perturb corners in greedy fashion to maximize IoU between the quadrilateral and the predicted pixels.

Step 1 helps remove any extraneous components (false positives) that were predicted as page regions. Some of these errors occur because the FCN makes local classification decisions. Similarly, Step 2 removes false negatives. In Step 3, we find a (rotated) bounding box for the main page region (OpenCV implementation [2]), but the bounding box encloses all predicted foreground pixels and is therefore sensitive to any false positive pixels that are outside the true page boundary. This bounding box is used to initialize the corners for iterative refinement in Step 4. At each refinement iteration, we measure the IoU of 16 perturbations of the corners (4 corners moved 1 pixel in 4 directions) and greedily update with the perturbation that has the highest IoU w.r.t. the FCN output. We stop the process when no perturbation improves IoU. Though this process is greedy and may terminate on a locally optimal quadrilateral, this rarely happened in our experiments because the FCN outputs are typically close to being quadrilateral shaped. Post-processing is done on 256x256 images, so there may be quantization artifacts after the quadrilaterals are upsampled to the original size.

5.3.3 PageNet Implementation Details

We implemented the FCN part of PageNet using the popular deep learning library Caffe [13]. The dataset used for training is detailed in Section 5.4. For preprocessing, color images are first resized to 256x256 pixels and pixel intensities are shifted and scaled to the range $[-0.5, 0.5]$. For ground truth, we label each pixel inside the image's annotated quadrilateral as foreground and all other pixels as background. The ground truth images are also resized to 256x256.

While all input images yield a probability map of the same size, we used 256x256 images in training and evaluation. While a larger input sizes could lead to slightly higher segmentation accuracy, we achieve good results with the computationally faster 256x256 size. Initial experiments with 128x128 inputs were less accurate.

To train the FCN, we used Stochastic Gradient Descent for 15000 weight updates with a mini-batch size of 2 images. We used an initial learning rate of 0.001, which was reduced to 0.0001 after 10000 weight updates. We used a momentum of 0.9, L2 regularization of 0.0005, and clipped gradients to have an L2 norm of 10. We trained 10 networks and used the validation set to select the best network for the results reported in Section 5.5.

5.4 Dataset

Our main dataset is the ICDAR 2017 Competition on Baseline Detection (CBAD) dataset [10], which are handwritten documents with varying levels of layout complexity (see Figure 5.1a,b,c). We combined both tracks of the competition data and separated the images into training, validation and test sets with 1635, 200, and 200 images respectively. We train PageNet on the training split of CBAD and evaluate on the validation and test splits of the same dataset.

We also evaluated on a subset of the CODH PMJT dataset[1], and on all images from the Saint Gall and Parzival datasets. The PMJT (Pre-Modern Japanese Text) dataset is taken from the Center for Open Data in the Humanities (CODH) [15] and consists of handwritten literature (see Figure 5.1d) and some graphics. We randomly sampled 10 pages from each the collections, excluding ID 20003967, to create an evaluation set of 140 images. The Saint Gall dataset [8] is a collection of 9th century manuscripts put together by the FKI: Research Group on Computer Vision and Artificial Intelligence. The Parzival dataset [9] is also put together by FKI and consists of 13th century Medieval German texts.

We also trained and evaluated PageNet on a private collection of Ohio death records[2], having a training set of 800 images, and validation and testing sets of 100 images each. This dataset, while relatively uniform in the types of documents, presents a unique challenge of overlay documents (see Figure 5.1e). While much of the document underneath an overlay is visible, much is still occluded, thus it is most desirable to localize just the overlay, which is a task that has not received much attention in the literature. These overlays represent approximately 12% of the images in the dataset.

5.4.1 Ground Truth Annotation

Our ground truth annotations consist of quadrilaterals encompassing the pages fully present in an image, not including any partial pages or page edges (if possible). We chose to use quadrilaterals rather than pixel level annotations as most pages are quadrilateral-shaped, and it is much faster to annotate polygon regions than pixel regions. Regions were manually annotated using an interface where the annotator clicks on each of the four corner vertices. In the case of multiple full pages, the quadrilateral encloses all pages present regardless of their orientation to each other. This typically occurs when two pages of a book are captured in a single image.

[1] `http://codh.rois.ac.jp/char-shape/`

[2] Data provided by FamilySearch

In order to give an upper bound on expected automated performance, we measured human agreement on the quadrilateral regions of our datasets. A second annotator provided region annotations for validation and test sets. To measure human agreement, this second set of regions was treated the same as the output of an automated system and scored w.r.t. the first set of annotations.

5.5 Results

In this section, we quantitatively and qualitatively compare PageNet with baseline systems and with human annotators. To evaluate system performance, we use Intersection over Union (IoU) averaged over all images in a dataset (mean IoU). We chose mIoU as our metric because it is commonly used to evaluate segmentation tasks.

5.5.1 Baselines systems

We compare PageNet with three baseline systems: full image, mean quadrilateral, and GrabCut [18]. For the full image baseline, the entire image is predicted as the main page region.

The mean quadrilateral can be computed as

$$\bar{x}_n = \frac{1}{N}\sum_{i=1}^{N}\frac{x_{in}}{w_i} \quad \bar{y}_n = \frac{1}{N}\sum_{i=1}^{N}\frac{y_{in}}{h_i} \tag{5.2}$$

where the mean quadrilateral is $(\bar{x}_1, \bar{y}_1, \ldots, \bar{x}_4, \bar{y}_4)$, the ith annotated quadrilateral is $(x_{i1}, y_{i1}, \ldots, x_{i4}, y_{i4})$, N is the number of training images, w_i and h_i are respectively the width and height of the ith image. The predicted quadrilateral for this baseline only relies on the height and width on the test image. For the jth image, the prediction is $(w_j\bar{x}_1,\, h_j\bar{y}_1, \ldots, w_j\bar{x}_4, h_j\bar{y}_4)$. Our mean quadrilateral was computed from the CBAD training split.

For the GrabCut baseline, we used the implementation in the OpenCV library [2]. For an initial object bounding box, we include the whole image except for a 5 pixel wide border

Dataset	# Images	PageNet (pixels)	PageNet (quads)	Full Image	Mean Quad	GrabCut	Human Agreement
CBAD-Train	1635	0.968	**0.971**	0.823	0.883	0.900	-
CBAD-Val	200	0.966	**0.968**	0.831	0.891	0.906	0.978
CBAD-Test	200	0.972	**0.974**	0.839	0.894	0.916	0.983
PMJT	140	0.936	**0.943**	0.809	0.897	0.904	0.982
Saint Gall	60	0.975	**0.987**	0.722	0.809	0.919	0.993
Parzival	47	0.956	**0.962**	0.848	0.920	0.925	0.989

Table 5.1: mIoU of PageNet and baseline systems. All rows used the same PageNet model trained on CBAD-train. PageNet (pixels) is the pixel segmentation of our proposed method taken after step 2 of post-processing (see Section 5.3.2 for details).

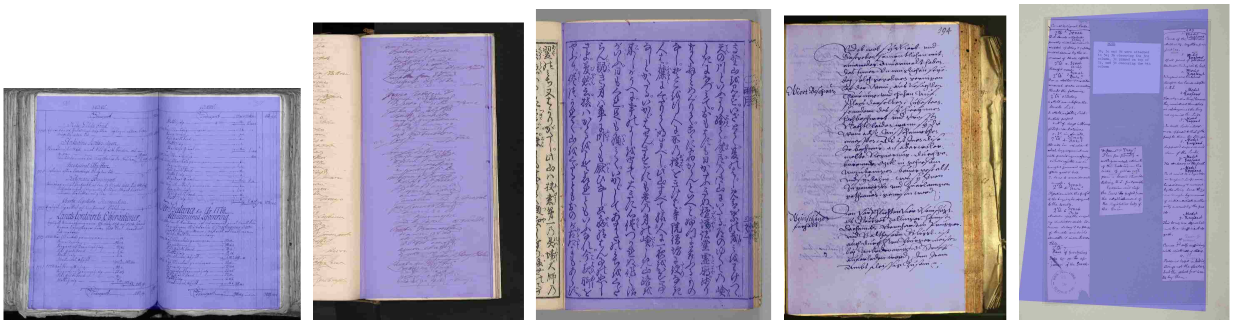
around the image edges. Unlike other baselines and the full PageNet results, GrabCut outputs a pixel mask (i.e., not a quadrilateral). Therefore, it is directly comparable to PageNet before we extract a quadrilateral.

We do not compare with methods designed to find the page frame as our method is designed to detect the entire main page region, including background regions.

5.5.2 Overall Results

We trained PageNet on CBAD-train and tested it and the baseline methods on 4 datasets of historical handwritten documents. Numerical results are shown in Table 5.1, and some example images are shown in Figure 5.4. For all datasets, except CBAD-train, we manually annotated each image twice in order to estimate human agreement on this task.

On all datasets, the full PageNet system performed the best of all automated systems and strongly outperformed the baseline methods. Notably, outputting quadrilaterals improves the pixel segmentation produced by the FCN, which shows that outputting a simpler region description does not decrease segmentation quality. There is little difference in the results for the different splits of CBAD, which indicates that PageNet does not overfit the training images. Performance is highest on Saint Gall because the border noise is largely limited to a black background. On PMJT, PageNet performed worst and most errors can be attributed



(a) (b) (c) (d) (e)

Figure 5.4: Example segmentations produced by PageNet. While (a)-(d) show correct segmentations on challenging documents, (e) shows a failure case due to the presence of multiple document snippets.

to incorrectly identifying page boundaries between pages, perhaps because the Japanese text is vertically aligned.

The full image baseline performs worst as it simply measures the average normalized area of the main page region. With the exception of Saint Gall, GrabCut only marginally outperforms the mean quadrilateral. As GrabCut is based on colors and edges, it often fails to exclude partial page regions (e.g., Figure 5.2) and sometimes labels dark text regions the same as the dark background. A few images in CBAD are well cropped and contain only the main page region, which is problematic for GrabCut because it will always attempt to find two distinct regions. In contrast, once trained, PageNet can classify an image as entirely the main page region if it does not contain border noise.

5.5.3 Comparison to Human Agreement

The last column of Table 5.1 shows the performance of a second annotator scored w.r.t. the original annotations. This performance captures the degree of error, or ambiguity, inherent with human annotations on this task, a level of performance which would be difficult for any automated system to surpass. For CBAD, PageNet is roughly 1% below human agreement, which indicates the network's proficiency at the task. On the PMJT dataset there is a larger

Dataset	# Images	FCN (pixels)	FCN (quads)
Ohio-Train	800	0.973	0.979
Ohio-Val	100	0.970	0.977
Ohio-Test	100	0.967	0.976

Table 5.2: mIoU results of PageNet trained on Ohio death certificates.

gap between automated and human performance and indicates that there is still room for improvement.

The human agreement results in Table 5.1 show the agreement between two annotators. We also measured the agreement of the same annotator labeling the same images on a different day. The same-annotator mIOU are 99.0% and 98.4% for CBAD-test and CBAD-val respectively. These are slightly higher than the mIOU of 98.3% and 97.8% obtained by a different annotator. This highlights the inherit ambiguity in labeling the corners of the main page region.

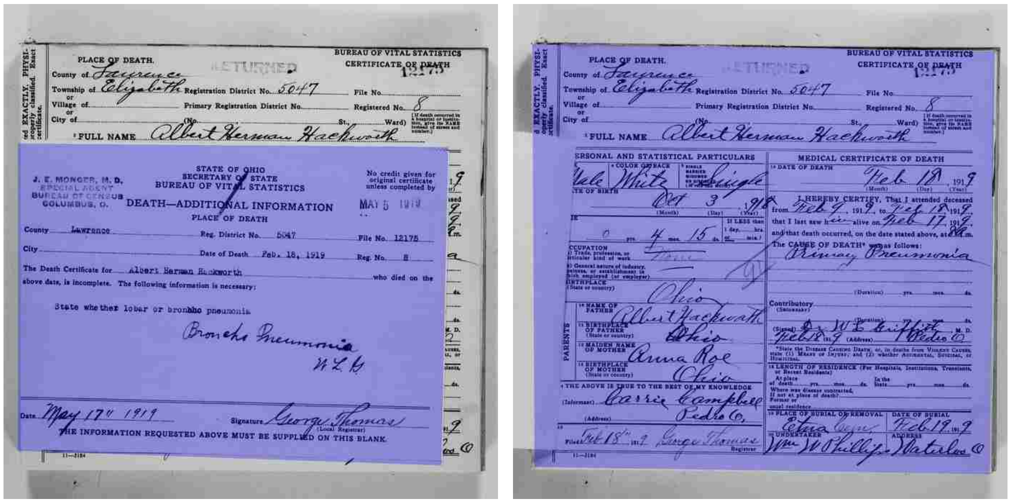
5.5.4 Overlay Performance

We also trained PageNet on a private dataset of Ohio death records. This dataset has several images where one document is overlayed on top of another document (e.g., Figure 5.1e), which creates particularly challenging textual noise. Table 5.2 shows results on this dataset.

In Figure 5.5, we show predicted segmentation masks for two images, which together show that PageNet correctly segments the overlayed image when present. Figure 5.5a contains a document overlayed on top of the document shown in Figure 5.5b. With the overlayed document, PageNet segments only the overlayed document, but when the overlayed document is removed, it segments the document underneath from the background.

5.6 Conclusion

We have presented a deep learning system, PageNet, which removes border noise by segmenting the main page region from the rest of the image. An FCN first predicts a class for each



(a) Overlayed

(b) Document Beneath

Figure 5.5: PageNet predictions for overlayed document. The document underneath the overlay in (a) is the same document without the overlay in (b)

input pixel, and then a quadrilateral region is extracted from the output of the FCN. We demonstrated near human performance on images similar to the training set and showed good performance on images from other collections. On an additional collection, we showed that PageNet can correctly segment overlayed documents.

References

[1] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 105–112 vol.1. IEEE, 2001. doi: 10.1109/ICCV.2001.937505.

[2] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel. Border noise removal of camera-captured document images using page frame detection. In *International Workshop on Camera Based Document Analysis and Recognition (CBDAR)*, pages 126–137. Springer, 2011.

[4] Arpita Chakraborty and Michael Blumenstein. Marginal noise reduction in historical handwritten documents–A survey. In *Document Analysis Systems (DAS)*, pages 323–328. IEEE, 2016.

[5] Arpita Chakraborty and Michael Blumenstein. Preserving text content from historical handwritten documents. In *Document Analysis Systems (DAS)*, pages 329–334. IEEE, 2016.

[6] Kai Chen, Mathias Seuret, Jean Hennebert, and Rolf Ingold. Convolutional neural networks for page segmentation of historical document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 965–970. IEEE, 2017.

[7] Kuo-Chin Fan, Yuan-Kai Wang, and Tsann-Ran Lay. Marginal noise removal of document images. *Pattern Recognition*, 35(11):2593–2611, Elsevier, 2002.

[8] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. Transcription alignment of Latin manuscripts using hidden markov models. In *International Workshop on Historical Document Imaging and Processing (HIP)*, pages 29–36. ACM, 2011.

[9] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7): 934–942, Elsevier, 2012.

[10] Tobias Grüning, Roger Labahn, Markus Diem, Florian Kleber, and Stefan Fiel. READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents. In *Document Analysis Systems (DAS)*, pages 351–356. IEEE, 2018.

[11] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025. 2015.

[12] L. Jagannathan and C.V. Jawahar. Perspective correction methods for camera based document analysis. In *International Workshop on Camera Based Document Analysis and Recognition (CBDAR)*, pages 148–154, 2005.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. In *International Conference on Computer Vision (ICCV)*, volume 259, page 268. IEEE, 1987.

[15] Asanobu Kitamoto. Release of PMJT character shape dataset and expectation for its usage. In *Future of Machine Recognition and Human Transcription*, 2 2017. doi: 10.20676/00000004.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 3431–3440. IEEE, 2015.

[17] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, pages 191–198. ACM, 1995.

[18] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, pages 309–314, 2004. doi: 10.1145/1186562.1015720.

[19] Faisal Shafait and Thomas M. Breuel. A simple and effective approach for border noise removal from document images. In *International Multitopic Conference (INMIC)*, pages 1–5. IEEE, 2009.

[20] Faisal Shafait, Joost Van Beusekom, Daniel Keysers, and Thomas M. Breuel. Document cleanup using page frame detection. *International Journal on Document Analysis and Recognition*, 11(2):81–96, Springer, 2008.

[21] Nikolaos Stamatopoulos, Basilios Gatos, and Thodoris Georgiou. Page frame detection for double page document images. In *Document Analysis Systems (DAS)*, pages 401–408. ACM, 2010.

[22] Chris Tensmeyer and Tony Martinez. Document image binarization with fully convolutional neural networks. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 99–104. IEEE, 2017.

[23] Godfried T. Toussaint. Solving geometric problems with the rotating calipers. In *Melecon*, volume 83, page A10. IEEE, 1983.

[24] Shih-Jui Yang, Chian C. Ho, Jian-Yuan Chen, and Chuan-Yu Chang. Practical homography-based perspective correction method for license plate recognition. In

International Conference on Information Security and Intelligence Control (ISIC), pages 198–201. IEEE, 2012.

[25] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, pages 1529–1537. IEEE, 2015.

Chapter 6

Physics Inspired Keypoint Regression in Convolutional Neural Networks

Abstract

In keypoint regression, models are trained to consume an image and produce the x,y coordinates of some entity (e.g. a person's nose). Inspired by the physical concept of an object's center of mass, we propose two spatially-aware and differentiable regression functions that compute continuous x,y coordinates from spatially discrete heatmap predictions. This allows end-to-end training of convolutional neural networks with dense output to learn keypoint locations. We demonstrate that our approach outperforms both strong baselines and a state-of-the-art regression function in synthetic experiments and on 5 real world keypoint regression datasets including faces, human pose, fashion, and document images.

In preparation for submission to *International Conference on Document Analysis and Recognition (ICDAR).*
Chris Tensmeyer and Tony Martinez. "Physics Inspired Keypoint Regression in Convolutional Neural Networks."

6.1 Introduction

Many challenging image tasks, such as object detection [24, 25], object segmentation [5], face alignment [28], human pose estimation [1], and text line detection [21] have recently been formulated as keypoint regression problems. The input to the model is an image and the desired output is the (x, y) coordinates of some specific entity in the image. Some common keypoint tasks include predicting facial features such as a person's nose and eyes as well as human joints like wrists, elbows, or shoulders. Deep convolutional neural networks (CNN) are state-of-the-art models in many image tasks are the primary choice for keypoint regression models.

CNN approaches to point prediction fall into three general categories: global regression, dense prediction, and hybrid methods.

- Direct regression methods jointly learn a regression model and non-linear neural network features to directly regress (x, y) keypoints under some distance based loss.
- Dense prediction or heatmap methods learn a confidence value for each pixel and the detected point is simply the integer coordinates of the pixel with highest confidence.
- Hybrid methods split the image into (perhaps overlapping) blocks, and for each block, the model outputs a confidence value and coordinates relative to the block location.

While these approaches have been shown to be effective, there are some draw backs. Direct regression methods, such as linear regression, typically ignore the spatial aspect of images and therefore must learn redundant sets of parameters to model the local appearance of each keypoint in different image locations. Thus these models lack translation equivariance (shifting the input correspondingly shifts the output), which is important if the keypoint can appear anywhere in the image. To deal with this issue, prior works have employed complicated cascaded models, where the first model performs a rough localization so that subsequent models do not need as much translation equivariance [28].

Dense prediction methods typically employ a Fully Convolutional Network (FCN) to output a confidence score for each input pixel. While this method is translation equivariant, there is a difference between the training objective and model inference, meaning training is not end-to-end [27]. Inference in these models is usually done by finding the maximum value in the dense heatmap and outputting the integer coordinates of that location. This limits the loss functions that can be used, as the loss can only instruct the network which pixels should have high or low confidence. In particular, the distance between predicted and ground truth keypoints cannot be used to train such models.

Hybrid methods, to some degree, have similar issues to dense predictions due to predicting block confidences, and they require selecting the size and degree of overlap of blocks. Hybrid methods are most commonly used for predicting a variable number of semantically equivalent keypoints, such as object bounding box corners in cluttered scenes. In contrast, our work focuses on tasks with a fixed number of keypoints, though our proposed method could be applied to the hybrid paradigm.

In this work, we propose two new neural network layers for keypoint prediction that overcome the limitations of both direct regression and dense prediction. These layers take as input a dense feature map of any size and output a continuous (x, y) coordinate at the center of high activations in the feature map. Taking inspiration from simple physics, the first layer interprets the input feature map as a grid of point masses and computes the physical *Center of Mass* (CoM). Though the masses are arranged on a discrete grid, the output CoM can be any continuous (x, y) point. Though this CoM layer yields good predictions, it has a slight deficiency in that the gradient is highest for points far away from the predicted (x, y) output. We fix this problem with our proposed *Median of Mass* (MoM) layer and demonstrate improved performance.

The CoM and MoM layers have some desirable properties. Both layers transform a heatmap-like input into a keypoint location. Thus, convolution layers can focus on learning a local appearance model rather than translating local features into a global (x, y) pair. In contrast to dense prediction methods, CoM and MoM have no separate inference procedure and both layers are differentiable, allowing for distance based loss functions to be used. Being differentiable also allows for latent point prediction for models utilizing spatial attention mechanisms used in, e.g., sequential image generation [9] and image captioning [34]. Finally, the layers are equivariant to similarity transforms, so shifting, scaling, or rotating the layer input produces a corresponding transformation in the output coordinates. This is useful for when keypoint appearance varies in location, size, and orientation.

We validate our proposed layers using synthetic data and 5 real datasets taken from the domains of faces, human pose, fashion, and document images. The MoM layer reduces Root Mean Squared Error by 28% and 30% over linear regression and dense prediction baselines. We also demonstrate the same or better performance compared with a recent state-of-the-art regression function introduced for human pose estimation [27] and show that CoM and MoM layers yield sparser and more concentrated detection heatmaps. Additionally, we exceed the previous best result on the READ-BAD corner detection task, achieving 98.1% intersection over union vs the previous 97.4%.

6.2 Background and Related Work

Many improvements to point regression have been made in the context of detecting facial keypoints for facial alignment and detecting joints for human pose estimation. Prior works can be categorized into four areas: architecture, structure prediction, loss functions, and regression functions. The architecture improvement category can be further split into direct regression architectures and dense prediction architectures. Our proposed work is a new regression function that performs direct regression from a dense prediction and thus spans both paradigms. Thus, improvements in architectures, structure prediction, and loss functions can be viewed as complimentary to our proposed layer.

6.2.1 Direct Regression

Direct regression architectures use a CNN to model

$$\mathbf{p} = f(I; \theta) \tag{6.1}$$

where $\mathbf{p} \in \mathbb{R}^{2k}$ are the predicted 2D image coordinates for k keypoints, I is the input image, and θ are the parameters of the model. Traditionally, the CNN is composed of several convolution layers, followed by fully connected layers. When there is no activation function after the final fully connected layer, the regression function is linear and can be formulated as

$$\mathbf{p} = \mathbf{W} f(I; \theta) \tag{6.2}$$

where $\mathbf{W}$ is a matrix of linear regression weights applied to the deep non-linear feature vector produced by f. Note that linear regression only incorporates spatial information through the features, rather than in the regression function itself. Designing a loss function for Eq. 6.1 or 6.2 is straightforward and the L2 loss is commonly used. Inference is also simple as the predicted points are directly computed.

Direct regression is greatly improved by introducing a cascade of direct regressors, where only the first network in the cascade takes as input the whole image [28]. Subsequent CNNs in the cascade are specific to clusters of target points (e.g. head and shoulder joint positions), and take as input smaller patches centered around previous network predictions. Thus the first network in the cascade only needs to output a rough localization of each keypoint and subsequent networks need not be translation equivariant because the points are roughly centered in the input.

Sun et al. [28] used 3 levels of cascade with 3 CNNs in the first level and 10 CNNs each for the second and third levels. Within a level, multiple CNNs predict the same keypoint at multiple resolutions, and predictions are averaged to reduce variance. In [35], the cascade of networks is preceded by a tight bounding box prediction. Kumar et al. [16] refined the initial predictions by predicting offset vectors from deep local descriptors. The DeepPose system also employed a cascade of 3 CNNs, where the scale of image patches input to later models depended on the distance between predicted keypoints (e.g. inter-shoulder distance) [32]. Deep Fashion Alignment [20] combines a 3-CNN cascade with predicting intermediate pseudo-labels that are based on clusters of visible keypoints (e.g. upper-body fashion vs lower-body). [1] used a cascade in conjunction with a proposed robust loss function that excludes outliers from backpropagating errors.

6.2.2 Dense Prediction

Dense prediction models formulate point regression as

$$\mathbf{D} = f(I; \theta) \tag{6.3}$$

where $\mathbf{D} \in \mathbb{R}^{k \times H \times W}$ is a heatmap of dimension $H \times W$ for k keypoints. f is typically modeled as a FCN or an FCN combined with a graphical model. Inference in such models is typically

$$\mathbf{p_k} = \arg\max_{x,y} \mathbf{D_k} \tag{6.4}$$

Due to the argmax, dense prediction methods are subject to quantization error. The magnitude of this error depends on the ratio of the dimensions of I to $H \times W$, which depends on the amount of pooling in the FCN. The ground truth in $\mathbf{g_k} = (x_k, y_k)$ form is not directly used, but is first converted to a dense form $\mathbf{G} \in \mathbb{R}^{k \times H \times W}$ so that $\mathbf{D}$ can be trained to approximate $\mathbf{G}$.

Learning dense prediction models is a simpler task than learning direct regression models [23, 31]. This is because direct regression (without cascade) models global context while dense prediction methods model local context and backpropagates error based on local feedback. Most keypoints are defined by local appearance, and difficult cases, such as occlusion, that benefit from global context are often handled by a subsequent structured model that encodes information about the relationships among multiple keypoints [31].

In [23, 30], **G** is constructed to be the concatenations of k 2D Gaussians, each centered around g_k and having fixed variance. Alternatively, Bulat and Tzimiropoulos [3, 4] use a binary **G**, setting pixels around g_k to 1 and train the CNN as a per-pixel classifier using a cross entropy loss. In [5], a similar scheme is used to sequentially predict vertices in a recurrent network for object segmentation.

Mass Displacement Networks use a differentiable geometric voting process, where the model outputs a confidence and a displacement vector for each pixel [22]. The evidence is smoothed according to a kernel and accumulated under a probabilistic noisy-or model to produce per-pixel probabilities. In [19], pixels vote for one of 50 log-polar bins for each keypoint. Inference is done by performing deconvolution with a fixed kernel to obtain per-pixel confidences.

6.2.3 Structure Prediction

Struture prediction methods refine initial keypoint predictions based on a spatial model of keypoint relationships. Convolutional Pose Machines predict an initial heatmap based on local appearance, and then iteratively update the heatmaps based on a recurrent contextual model [33]. Tompson et al. [31] use a fully connected Markov Random Field and model pairwise connections with learned convolutional priors. Sun et al. [26] regress differences in keypoint locations, which have lower variance than keypoint locations due to geometric constraints of human poses. Chen et al. [6] learn a prior over biologically plausible human poses using adversarial learning.

Structure prediction bears some similarity to cascade methods, but structure prediction methods learn features from the initial heatmaps. In contrast, the only information passed between cascade levels are estimated keypoints.

6.2.4 Regression Functions

Regression functions for image data proposed in the literature include mixture of inverse linear regressions for head pose estimation [17] and Support Vector Regressors (SVR) for facial beauty assessment [8]. Integral Pose Regression directly regresses (x, y) from latent dense predictions and is similar to our proposed CoM approach [27]. We explore this connection in Section 6.3. Hou et al. [12] demonstrate that learned Smooth Adaptive Activation Functions (SAAF) on the penultimate CNN layer leads to lower model bias error on age estimation and human pose estimation tasks.

6.3 Point Regression Layers

This section explains our two proposed layers for keypoint regression, which we have named the Center of Mass (CoM) and the Median of Mass (MoM) layers. Both layers take in a single feature map and produce a single pair of (x, y) coordinates. That is, they compute a function $f : \mathbb{R}^{H\times W} \to \mathbb{R}^2$, where H, W are the feature map dimensions. This can be applied to any number of input feature maps to produce the desired number of keypoint predictions.

6.3.1 Center of Mass Layer

This layer draws inspiration from physics by interpreting the input feature map as a 2D grid of point masses, where the mass of each point is determined by the corresponding feature map value. Under Newton's laws, such a system of masses may be abstracted as a single point, $\mathbf{c} = (c_x, c_y)$, whose mass is the total mass of the system. This point is known as the CoM of the system and satisfies

$$\sum_{ij} m_{ij}(i - c_x) = \sum_{ij} m_{ij}(j - c_y) = 0 \tag{6.5}$$

where m_{ij} is the value of the input feature map at position (i, j). We can interpret the CoM as a distance-weighted average of the masses, where masses farthest away from $\mathbf{c}$ have the most influence on the position of the CoM. Under the assumption that $\sum_{ij} m_{ij} \neq 0$, $\mathbf{c}$ is unique and can be computed by rearranging Eq. 6.5 to

$$c_x = \frac{\sum_{ij} i m_{ij}}{\sum_{ij} m_{ij}} \quad c_y = \frac{\sum_{ij} j m_{ij}}{\sum_{ij} m_{ij}} \tag{6.6}$$

The derivative is also simple to derive and compute. Starting with the x-coordinate, we have

$$\frac{dc_x}{dm_{uv}} = \frac{u \sum_{ij} m_{ij} - \sum_{ij} i m_{ij}}{(\sum_{ij} m_{ij})^2} = \frac{u - c_x}{\sum_{ij} m_{ij}} \tag{6.7}$$

Similarly, the y-coordinate is

$$\frac{dc_y}{dm_{uv}} = \frac{v - c_y}{\sum_{ij} m_{ij}} \tag{6.8}$$

From Eqs. 6.7 and 6.8, we can see that the gradient magnitude at location (u, v) is proportional to $||(u, v) - \mathbf{c}||_2$, so gradient based training will encourage input locations far from c to have high activations. This is undesirable because keypoints are often identifiable by local appearance and large activations far away from the keypoint can lead to imprecise localization. However, in practice, we find using a ReLU activation ($ReLU(x) = max(0, x)$) before the CoM layer helps alleviate this problem. The input locations far away from $\mathbf{c}$ tend to be set to 0 by ReLU and thus zero out the large CoM gradient. In practice, the FCN learns a local detector for keypoints (similar to dense prediction methods) as input to the CoM layer and has few non-zero activations far from the keypoint.

6.3.2 Median of Mass Layer

The MoM layer is similar to the CoM layer, but solves the distance weighting of gradients problem. While nature uses a linear distance weighting function in Eq. 6.5 to define a system's CoM, we are free to choose any non-decreasing function of distance to obtain a unique Generalized Center of Mass (GCoM). For the x-coordinate we have

$$\sum_{ij} m_{ij} \operatorname{sign}(i - c_x) g(|i - c_x|) = 0 \tag{6.9}$$

where g is a non-decreasing function. Without this restriction on g, Eq. 6.9 could have multiple solutions, leading to ambiguous or inconsistent layer output. If g is linear, Eq. 6.9 reduces to Eq. 6.5 and we get the vanilla CoM. If we choose g to be a constant function, so that masses are not distance weighted at all, Eq. 6.9 reduces to:

$$\sum_{ij} m_{ij} \operatorname{sign}(i - c_x) = \sum_{ij} m_{ij} \operatorname{sign}(j - c_y) = 0 \tag{6.10}$$

However, given the interpretation of the input matrix $\mathbf{M}$ as point masses, Eq. 6.10 likely has no exact solution. Therefore, we reinterpret the input to be piecewise constant function $m(i,j) = m_{\lfloor i \rfloor, \lfloor j \rfloor}$ and rewrite Eq. 6.10 as

$$\iint m(i,j) \operatorname{sign}(i - c_x)\, di\, dj = \iint m(i,j) \operatorname{sign}(j - c_y)\, di\, dj = 0 \tag{6.11}$$

To solve Eq. 6.11, we compute row and column sums of $\mathbf{M}$ and find the median values of these 1D vectors. While the median is not likely to fall on any integer position, we find the two positions around the median and perform linear interpolation to find the continuous median.

Let r be a vector of column sums ($r_i = \sum_j m_{ij}$), R be the corresponding vector of cumulative sums ($R_i = \sum_{k=1}^{i} r_k$), T be the total mass, and P be the normalized cumulative sum ($P_i = \frac{R_i}{T}$). We find the largest u, such that $P_u < 0.5$ and $r_u \neq 0$. Similarly, we find the smallest v such that $P_v \geq 0.5$ and $r_v \neq 0$. Note that we may have $v \neq (u+1)$ due to 0 values in r, though this is not commonly encountered in practice. Then we find the point where the line through (u, P_u) and (v, P_v) intersects the line $y = 0.5$. Defining $s = \frac{v-u}{P_v - P_u}$, we have $c_x = u + \frac{0.5 - P_u}{s}$.

We can determine c_y from the same process by using row sums instead of column sums. The backwards step for computing $\frac{\partial c_x}{\partial m_{ij}}$ proceeds as

$$\frac{\partial P_k}{\partial m_{ij}} = \begin{cases} \frac{T-P_k}{T^2} & \text{for } i \leq k \\ \\ \frac{-P_k}{T^2} & \text{for } i > k \end{cases} \tag{6.12}$$

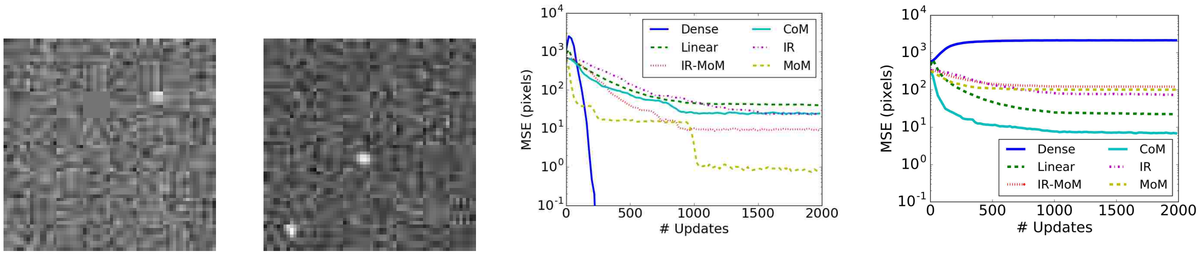
$$\frac{\partial s}{\partial m_{ij}} = \frac{\frac{\partial P_u}{\partial m_{ij}} - \frac{\partial P_v}{\partial m_{ij}}}{v-u} \tag{6.13}$$

$$\frac{\partial c_x}{\partial m_{ij}} = -\frac{s\frac{\partial P_u}{\partial m_{ij}} + \frac{\partial s}{\partial m_{ij}}(0.5 - P_u)}{s^2} \tag{6.14}$$

While Eq. 6.14 is not as simple as Eq. 6.7, the magnitude of $\frac{\partial c_x}{\partial m_{ij}}$ no longer depends on the distance between c_x and i. The gradient only depends on if $i \leq u$, $u < i \leq v$, or $i > v$. The gradient magnitude differences between these ranges are small because unless $\mathbf{M}$ is highly skewed, $T - P_u \approx T - P_v \approx -P_u \approx -P_v \approx -0.5$. The derivative within these three ranges is constant, which helps the MoM layer be robust to outlier activations and therefore learn to activate near the keypoint of interest.

Relation to Integral Regression

After developing and implementing our layers, we found a similar method in concurrent work called Integral Regression (IR) [27]. Our CoM layer computes $\mathbf{c} = f_{CoM}(ReLU(\mathbf{F_k}))$, while IR computes $\mathbf{c} = f_{CoM}(\sigma(\mathbf{F_k}))$, were $\mathbf{F_k}$ is the k^{th} feature map of the last convolutional layer and $\sigma(\cdot)$ is the Softmax function. Because $\sigma(\cdot)$ backpropagates non-zero values to all its inputs, input locations far away from $\mathbf{c}$ in IR can receive large backpropagated errors due to distance weighting of errors (Eqs. 6.7 and 6.8). This problem is addressed in our novel MoM layer and we show in Section 6.6 that using ReLU leads to sparser heatmap inputs. Though this issue is not explicitly addressed in [27], their complete model was trained with



(a) Task 1 input (b) Task 2 input (c) Task 1 results (d) Task 2 results

Figure 6.1: Example Synthetic task inputs (a,b) and results for each regression function (c,d). For (a,b) the images are scaled and shifted to the range [0,255] and the target value is indicated in red (best viewed in color).

an auxiliary L2 loss between $\mathbf{F}$ and 2D Gaussians centered around each $\mathbf{g_k}$, which encourages heatmaps to only activate around $\mathbf{g_k}$.

6.4 Experiments - Synthetic Data

In this section, we explore the performance of several regression functions on two synthetic point regression tasks. The first task simulates the output of a noisy, but accurate sliding window detector. The background of each 64x64 input is composed of i.i.d. Gaussian noise with $\mu = 0, \sigma = 0.2$. For each input, we sample random ground truth with integer coordinates, $\mathbf{g} = (g_x, g_y)$. We set nearby input pixels $\mathbf{p}$, such that $||\mathbf{g} - \mathbf{p}||_2 \leq 2$, to higher values using Gaussians with parameters $\mu = (1 - 0.2||\mathbf{g} - \mathbf{p}||_2), \sigma = 0.2$. An example input is found in Figure 6.1a.

The second task mimics the scenario where the local appearance of a keypoint is occluded or otherwise not detected, but the keypoint location can be inferred based on its relationship to two other detected points. The inputs for this task have the same background as in the first task, but instead have two foreground regions around two context points that we denote $\mathbf{g_1}, \mathbf{g_2}$. The pixels within radius 2 of either $\mathbf{g_1}$ or $\mathbf{g_2}$ are sampled from Gaussians with higher mean as in task 1. The ground truth for this task is the midpoint $\mathbf{g} = \frac{\mathbf{g_1}+\mathbf{g_2}}{2}$.

We compare CoM, MoM, IR [27], IR-MoM, Linear, and Dense regression functions using 2-layer networks with architectures detailed in Table 6.1. For the Dense architecture,

Table 6.1: Network architectures used in synthetic experiments. The first layer of all networks is a conv layer with 8 feature maps and 3x3 kernels followed by a ReLU activation. Note that Linear Reg. is a parameterized prediction, so all networks have 2 layers with learnable parameters. CE = per-pixel binary Cross Entropy.

Function	2nd Layer	Act.	Prediction	Loss
CoM	conv 3x3	ReLU	CoM	L2
MoM	conv 3x3	ReLU	MoM	L2
IR	conv 3x3	Softmax	CoM	L2
IR-MoM	conv 3x3	Softmax	MoM	L2
Linear	none	none	Linear Reg.	L2
Dense	conv 3x3	Sigmoid	Argmax	CE

the target value is a binary mask with pixels within distance 2 of **g** set to 1 and all other pixels set to 0. For all other networks, the target value is **g** itself.

For each regression function, we selected the best initial learning rate from {0.1, 0.03, 0.01, 0.003, 0.001} and trained 100 networks from different random initializations. We trained each network for 2000 weight updates using Stochastic Gradient Descent (SGD) and recorded the Mean Squared Error (MSE) in pixels between the prediction and the ground truth for each input mini-batch of 16 instances. We use momentum of 0.9, L2 weight decay of 0.0005, and after 1000 updates, we divide the learning rate by 10. After uniform smoothing for viewing purposes, we report the average MSE of each method as a function of training iteration.

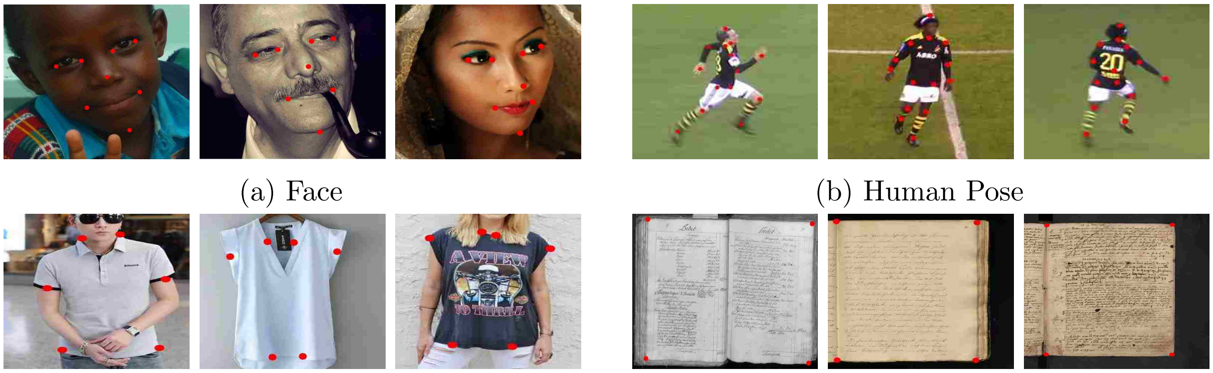
Figures 6.1c and 6.1d show the results on task 1 and 2 respectively. On task 1, Dense prediction quickly converges to near 0 MSE because the argmax of the input is already a near perfect solution. MoM is the next best method, converging to a MSE of approximately 1 pixel, followed by IR-MoM with a MSE just over 10. CoM and IR converge to similar MSE because high value background pixels produce small but non-zero inputs to the CoM regression function and therefore skew the prediction due to the distance weighting expressed in Eq. 6.5. MoM and IR-MoM are more robust to this because far away outliers are not distance weighted. Linear regression converges slowly because regression weights are specific to each image location, so it must see many instances at each image location to make correct predictions.

On task 2, CoM performs best because it zeros out background pixels, leaving only high activations around the two context points. In contrast, IR and IR-MoM put the majority of their heatmap activation on the two context points, but the many small, but non-zero activations skew the prediction away from the midpoint of the two context points. Surprisingly, MoM fails to filter out many of the background pixels, which introduces skewed predictions when these background pixels are not symmetrically arranged about the target value. Dense prediction cannot learn the task because local features do not identify the target value. Linear regression performs better than many other methods because it combines information from $\mathbf{g_1}, \mathbf{g_2}$ on a global level.

We see that some methods are better at modeling local appearance than modeling keypoint context. In real world data, we observe from the heatmap activations that each method attempts to do model both and that there should be a trade-off between the two based on the specifics of the image domain.

6.5 Experiments - Real Data

In this section, we validate our two proposed regression functions, CoM and MoM, using real world data. Rather than focus on a single domain, we aim to show that our methods work well on datasets from a variety of image domains, including faces, human pose, fashion, and document images. We follow a similar protocol to that used in Section 6.4, comparing six regression functions using identical network architectures. Additionally, each regression function was trained with two distance based loss functions: L1 and Tukey bi-weight loss [1]. The Tukey loss normalizes output errors based on a robust per-dimension variance measure and yields no error gradient on detected outliers. Initial experiments (data not shown) demonstrated that L2 loss did not perform as well as L1 loss. For Dense prediction, per-pixel cross entropy loss is used.



(a) Face

(b) Human Pose

(c) Fashion

(d) Document

Figure 6.2: Example images from domains used for evaluation. Each image is labeled with ground truth keypoints in red.

6.5.1 Datasets

We use datasets from four domains: faces, human pose, fashion, and document images (Figure 6.2). For faces, we used the Helen [18] and LFPW [2] datasets. Because we are interested in evaluating regression functions rather than producing a complete face alignment system that involves face detector, cascade, and structure prediction components, we preprocessed the images to remove background and non-annotated faces and restrict prediction to 8 facial landmarks. The preprocessing involved finding face bounding boxes using the ground truth and padding this bounding box on each side by a random amount between 10% and 25% of the box width and height. The 8 facial landmarks we predict can be uniquely identified by local appearance and are the corners of both eyes, the corners of the mouth, the tip of the nose, and the bottom of the chin. Helen is split into 1800/200/330 images for train/val/test, and LFPW is split into 711/100/224.

We used the 2D Football dataset [14] and all 14 joint positions for human post estimation. Images already have the person in the center of the image, so no preprocessing was necessary. The joints to be regressed include the left/right foot, knee, hip, hand, elbow, and shoulder, as well as the bottom and top of the head. This dataset is split into 3600/300/2007 for train/val/test respectively.

For fashion we used a subset of the large DeepFashion dataset [20]. Each image in DeepFashion is categorized into a clothing type (upper, lower, or full body). We selected variation 1 of the upper body clothes and filtered out images with occluded landmarks, which left 5607 images split into 3794/917/896 for train/val/test. As with face datasets, we perform a randomly padded crop around the annotated item of clothing because some images have multiple items of clothing. The 6 keypoints are left and right collars, sleeves, and hems.

In the document domain, we predict the 4 corners of pages using the READ-BAD images [10] and the corner annotations provided in [29]. Finding the corners allows marginal noise such as book edges and partial adjacent pages (see Figure 6.2d) to be removed before further processing This dataset is split into 1635/200/200 for train/val/test and needed no cropping.

6.5.2 Network Architecture

Our main architecture, used for face, pose, and fashion datasets, is a 10-layer encoder-decoder network that downsamples by a factor of 16 in the 5 encoder layers using max pooling. Layers 6-9 upsample the feature maps by a factor of 2 each using bilinear interpolation. The last layer is a 1x1 convolution followed by the chosen regression function. This produces a dense output the same size as the input, which is appropriate for CoM, MoM, IR, IR-MoM, and Dense functions. However, Linear Regression does not need a dense output, so we modify the architecture to extend the encoder portion, remove the decoder, and add 3 fully connected layers. All layers but the last apply Batch Normalization [13] and ReLU activation.

For READ-BAD, we extend the branching multi-scale network used in [29] to have a depth of 10. To utilize the domain knowledge that the 4 page corners occur in separate quadrants of the image, we spatially split the feature maps at layer 7 into 4 parts. Each quadrant is fed to 3 distinct convolution layers to regress each corner separately. Diagrams and full specification of network architectures can be found in the supplementary material.

6.5.3 Evaluation Protocol

For each domain, we first select a LR $\in \{0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001\}$ for each regression function and loss pair. This is done by training 4 networks for 5000 iterations for each combination and selecting the LR with the lowest average MSE. Then 10 networks are trained for each combination for 35000 iterations at the selected learning rate, measuring validation MSE every 100 iterations. The output of each training procedure is the set of best performing weights on the validation set. Of the 10 trained networks for each function/loss, the best validation network is selected to obtain our test set metrics.

While each domain has its own prefered metrics, which we use in domain-specific discussion, we compare across domains using the Root-MSE (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_i^N ||\mathbf{g_i} - \mathbf{p_i}||_2^2}{NK}} \tag{6.15}$$

where $\mathbf{g_i} \in \mathrm{R}^{2K}$ is the ground truth for instance i, $\mathbf{p_i}$ is the predicted vector of keypoint locations, and N is the number of instances. The RMSE can roughly be interpreted as average deviation from the ground truth. Because we use normalized coordinates in $[0, 1]$, a RMSE of 0.02 corresponds to an average deviation of 2% of the image dimension or ≈ 5 pixels for a 256x256 image.

Training Details

Each network was trained from random initialization [11] for 35000 iterations using the ADAM optimizer [15] with default parameters. We used L2 weight decay of 0.0005 and divided the LR by a factor of 10 every 15000 iterations. Inputs are stochastically rotated and padded. LFPW networks were only trained for 25000 iterations due to a smaller training set, and only 5 READ-BAD networks were trained for each function/loss due to longer training time.

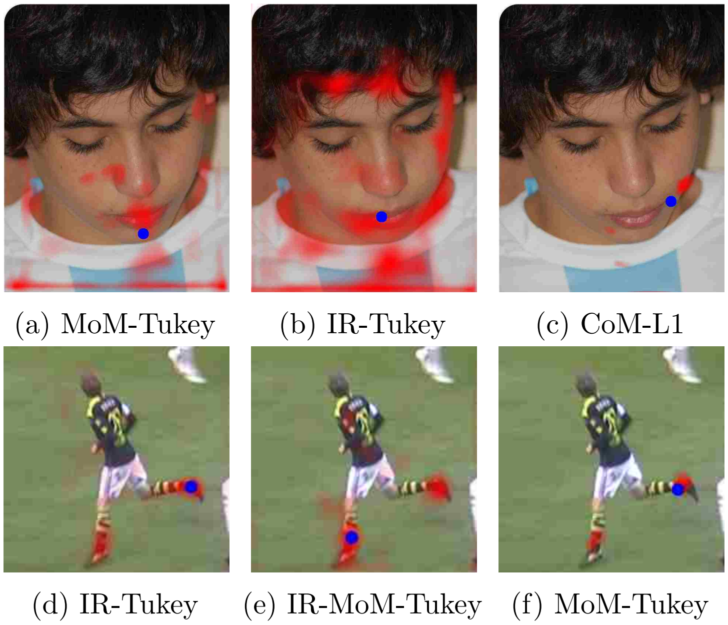
Table 6.2: RMSE results for each regression function and loss combination on 5 datasets. CoM, MoM, IR-MoM are our methods, IR is from [27], and Linear/Dense are baseline methods.

Function	Loss	Helen	Football	READ-BAD	LFPW	Fashion	Avg Rank
CoM	L1	**0.0148**	0.0348	0.0222	**0.0195**	0.0623	3.0
MoM	L1	0.0170	**0.0342**	0.0132	0.0219	**0.0587**	**1.8**
IR-MoM	L1	0.0172	0.0346	**0.0120**	0.0235	0.0598	2.4
IR	L1	0.0207	0.0346	0.0132	0.0303	0.0618	3.4
Linear	L1	0.0207	0.0367	0.0161	0.0209	0.0750	4.4
Dense	CE	0.0178	0.0373	0.0241	0.0339	0.0638	5.4
CoM	Tukey	0.0149	0.0319	0.0271	0.0203	0.0608	3.6
MoM	Tukey	**0.0138**	0.0319	0.0132	**0.0163**	0.0548	**2.0**
IR-MoM	Tukey	0.0149	0.0306	**0.0106**	0.0182	0.0554	2.2
IR	Tukey	0.0147	**0.0293**	0.0124	0.0218	**0.0485**	**2.0**
Linear	Tukey	0.0251	0.0347	0.0167	0.0230	0.0730	4.8

6.6 Results and Discussion

Our main results for all datasets are presented in Table 6.2. MoM has the best average rank for both the L1 and Tukey losses, though it does tie with IR-Tukey. MoM does particularly well on face images, while IR-Tukey does best on human pose and the closely related fashion domain. This is likely because MoM focuses more on modeling local appearance through sparse heatmaps and because facial landmarks have more distinct local appearance than human joints. For example, determining which leg is left or right depends on if the image is a frontal or dorsal view of the person. Because IR backpropagates error through the whole heatmap, it learns a better model of keypoint context. Overall, with the exception of the fashion domain, we see that MoM either outperforms or is complimentary to IR.

MoM-Tukey strongly outperforms the baseline measures of Linear Regression and Dense prediction and comparatively reduces RMSE by 28% and 30% respectively. On most datasets, MoM outperforms CoM, suggesting that eliminating the distance weighting of gradients (Eq. 6.7) leads to improved solutions. We also observe that networks trained with Tukey loss outperform corresponding L1 networks on all datasets except READ-BAD. We believe this is due to smaller variance in the corner locations for this dataset.



(a) MoM-Tukey (b) IR-Tukey (c) CoM-L1

(d) IR-Tukey (e) IR-MoM-Tukey (f) MoM-Tukey

Figure 6.3: Comparison of learned heatmaps on difficult images using 3 best networks on Helen (a-c) and Football (d-e). The blue dot indicates the predicted location. For Helen, the keypoint is the chin. For Football, it is the left foot (the one on the ground). CoM and MoM have sparser and more concentrated heatmaps and therefore have less noisy predictions than IR based methods.

On both face datasets, MoM-Tukey had the lowest RMSE overall and CoM-L1 was best performing of all networks trained with L1 loss. MoM-Tukey was able to localize the most difficult facial keypoint, the chin, much better than other methods. On chins, MoM-Tukey achieved 0.034 RMSE on Helen vs 0.037 for IR-Tukey and 0.033 on LFPW vs 0.041 for IR-MoM-Tukey.

In general, MoM and CoM networks learned to have more sparse and concentrated heatmap inputs to the regression function (see Figure 6.3). This is because the ReLU activation in MoM and CoM encourages distant heatmap entries to be 0, whereas the Softmax used in IR methods sets all heatmap values to be positive.

The IR methods performed best on human pose estimation, which is not surprising given that IR was developed for this problem. While IR-Tukey has the lowest RMSE, IR-MoM-Tukey has the highest Percentage of Correctly estimated Parts (PCP) score [7]. PCP counts a point as correctly localized if the predicted point is within 50% of the distance between adjacent ground truth points. IR-Tukey has an overall PCP score of 89.4%, while IR-MoM-Tukey has a score of 90.0%. The PCP difference is greatest for forearms (72.0% vs 74.4%). This suggests that MoM and IR are complimentary methods for some domains.

As with faces, the IR-Tukey methods do not produce sparse heatmaps even though they are the best performing method (see Figure 6.3d-f). This suggests that further improvement might be made through sparsification: e.g. setting small values in the heatmap to 0 after the Softmax.

As with the human pose task, MoM was the best performing L1 trained method and IR was the best Tukey method. This is likely because the fashion domain is quite similar to human pose, though there is more variation in where keypoints can be located (e.g. short sleeve vs long sleeve) [20]. This makes it more critical to model a large context, which is one of advantage of IR over MoM. The overall RMSE for this dataset is higher, which reflects the greater ambiguity in the keypoint definitions.

In [29], the evaluation metric was the Intersection over Union (IoU) of the predicted quadrilaterals and the ground truth quadrilaterals defined by the locations of the 4 corners. Our best method, IR-MoM-Tukey achieves an IoU of 98.1%, compared to 97.4% for the best previously published result and compared to 98.3% for human agreement [29].

6.7 Conclusion

We have presented two novel regression function neural network layers, CoM and MoM, that predict continuous (x, y) coordinates from dense heatmap representations. Using synthetic and real data, we have demonstrated that CoM and MoM methods significantly outperform linear regression and dense prediction baseline methods, reducing RMSE by up to 30%. Additionally, we have compared to the state-of-the-art Integral Regression (IR) method proposed in concurrent work and found that MoM outperforms IR when trained with L1 loss. As well, our proposed layers produce sparser and more concentrated heatmap predictions compared to IR. On the READ-BAD corner detection task, we exceeded the previous state-of-the-art result, reducing IOU error by 27% and achieving near human performance.

References

[1] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *International Conference on Computer Vision (ICCV)*, pages 2830–2838. IEEE, 2015.

[2] Peter N. Belhumeur, David W. Jacobs, David J. Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. *Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2930–2940, IEEE, 2013.

[3] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision (ECCV)*, pages 717–732. Springer, 2016.

[4] Adrian Bulat and Yorgos Tzimiropoulos. Convolutional aggregation of local evidence for large pose face alignment. In *British Machine Vision Conference (BMVC)*, pages 86.1–86.12. British Machine Vision Association, September 2016.

[5] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5230–5238. IEEE, July 2017.

[6] Yu Chen, Chunhua Shen, Hao Chen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial learning of structure-aware fully convolutional networks for landmark localization. *Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2019.

[7] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2008.

[8] Junying Gan, Lichen Li, Yikui Zhai, and Yinhua Liu. Deep self-taught learning for facial beauty prediction. *Neurocomputing*, 144:295–303, Elsevier, 2014.

[9] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *International Conference on Machine Learning (ICML)*, pages 1462–1471. JMLR, 2015.

[10] Tobias Grüning, Roger Labahn, Markus Diem, Florian Kleber, and Stefan Fiel. READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents. In *Document Analysis Systems (DAS)*, pages 351–356. IEEE, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *International Conference on Computer Vision (ICCV)*, pages 1026–1034. IEEE, 2015.

[12] Le Hou, Dimitris Samaras, Tahsin Kurc, Yi Gao, and Joel Saltz. ConvNets with smooth adaptive activation functions for regression. In *Artificial Intelligence and Statistics*, pages 430–439, 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML*, volume 37, pages 448–456. JMLR, 2015.

[14] Vahid Kazemi, Magnus Burenius, Hossein Azizpour, and Josephine Sullivan. Multi-view body part recognition with random forests. In *British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2013.

[15] Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *International Conference on Learning Representation (ICLR)*, ArXiv, 2014.

[16] Amit Kumar, Rajeev Ranjan, Vishal Patel, and Rama Chellappa. Face alignment by local deep descriptor regression. *arXiv preprint arXiv:1601.07950*, 2016.

[17] Stéphane Lathuiliere, Rémi Juge, Pablo Mesejo, Rafael Munoz-Salinas, and Radu Horaud. Deep mixture of linear inverse regressions applied to head-pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 3, pages 4817–4825. IEEE, 2017.

[18] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S. Huang. Interactive facial feature localization. In *European Conference on Computer Vision (ECCV)*, pages 679–692. Springer, 2012.

[19] Ita Lifshitz, Ethan Fetaya, and Shimon Ullman. Human pose estimation using deep consensus voting. In *European Conference on Computer Vision (ECCV)*, pages 246–260. Springer, 2016.

[20] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Fashion landmark detection in the wild. In *European Conference on Computer Vision (ECCV)*, pages 229–245. Springer, 2016.

[21] Bastien Moysset, Pierre Adam, Christian Wolf, and Jérôme Louradour. Space displacement localization neural networks to locate origin points of handwritten text lines in historical documents. In *International Workshop on Historical Document Imaging and Processing (HIP)*, pages 1–8. ACM, 2015.

[22] Natalia Neverova and Iasonas Kokkinos. Mass displacement networks. In *British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2018.

[23] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *International Conference on Computer Vision (ICCV)*, pages 1913–1921. IEEE, 2015.

[24] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271. IEEE, July 2017.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99. MIT Press, 2015.

[26] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. Compositional human pose regression. In *International Conference on Computer Vision (ICCV)*, volume 2. IEEE, 2017.

[27] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *European Conference on Computer Vision (ECCV)*, pages 529–545. Springer, 2018.

[28] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3476–3483. IEEE, 2013.

[29] Chris Tensmeyer, Brian Davis, Curtis Wigington, Iain Lee, and Bill Barrett. PageNet: Page boundary extraction in historical handwritten documents. In *International Workshop on Historical Document Imaging and Processing (HIP)*, pages 59–64. ACM, 2017.

[30] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 648–656. IEEE, 2015.

[31] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1799–1807. MIT Press, 2014.

[32] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660. IEEE, 2014.

[33] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732. IEEE, 2016.

[34] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057. JMLR, 2015.

[35] Erjin Zhou, Haoqiang Fan, Zhimin Cao, Yuning Jiang, and Qi Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *International Conference on Computer Vision (ICCV) Workshops*, pages 386–391. IEEE, 2013.

Chapter 7

Deep Splitting and Merging for Table Structure Decomposition

Abstract

Given the large variety and complexity of tables, table structure extraction is a challenging task in automated document analysis systems. We present a pair of novel deep learning models that given an input image, 1) predict the basic table grid pattern and 2) predict which grid elements should be merged to recover cells that span multiple rows and columns. These models utilize pooling regions designed for their respective subtasks. We achieve state-of-the-art performance on the public ICDAR 2013 Table Competition dataset of PDF documents. On a much larger private dataset which we used to train the models, we strongly outperform both a state-of-the-art deep model and a major commerical software system.

Submitted to *International Conference on Document Analysis and Recognition (ICDAR).* Chris Tensmeyer, Brian Price, Vlad Morariu, Scott Cohen, and Tony Martinez. "Deep Splitting and Merging for Table Structure Decomposition."

7.1 Introduction

Tables are commonly used to present structured data in documents. Automatically determining the structure of tables, i.e. the locations of cells and how they are organized into columns and rows, is a crucial part of document understanding and information extraction.

This work focuses on documents as images and in Portable Document Format (PDF) that contain tables with no structural annotations. Images, such as document scans, are composed of pixels that visually reproduce the document, but do not have any structural information. While PDFs are composed of discrete drawing commands for text and line art, table structure is not explicitly encoded in these commands. For example, a single text drawing command may be used to render part of a cell, an entire cell, parts of multiple cells, or multiple entire cells. Such ambiguities makes it non-trivial to extract table structure from PDF drawing commands.

Several factors contribute to the difficulty of parsing table structure (see [6] for an overview). Lined tables are generally easier than unlined tables [4] because they have explicit visual cues that segment cells, columns, and rows. Other tables are partially lined or use shading to denote boundaries. Tables may have a complex and heirarchical header structure or have no headers. Many tables contain multi-line cells, which introduces the difficulty of determining if vertically adjacent text lines are part of the same cell or not. Additionally, cells that span multiple columns and rows add to table complexity.

Many previous works that extract table structure use hand-crafted rules and heuristics to analyze the drawing commands in PDFs [4, 9, 10, 12]. Recently, deep learning has been proposed to learn the table structure directly from images [11], though this model was originally designed for semantic segmentation of natural scenes. While heuristic methods have previously outperformed learning methods on PDFs, they cannot be directly applied to images. Furthermore, heuristic rules have many parameters that must be tuned as tables vary widely in structure, visual appearance, and complexity.

In this work, we propose a pair of deep learning models that given a well-cropped table image, 1) predict the fine grid structure of the table and 2) predict where cells span multiple columns and rows. In the first model, information is pooled along rows and columns of pixels to help learned features propagate across large images. The output of this model is the grid structure of the table, ignoring any spanning cells. The second model takes as input the grid structure and the original table image and predicts which elements of the grid need to be merged to recover the spanning table cells.

We achieve state-of-the-art performance on the ICDAR 2013 Table Competition dataset [4], obtaining an adjacency relationship F-measure of 95.15% compared to the best published result of 94.6% [4]. We also evaluate on a much larger and more diverse private dataset of table images, greatly outperforming a previous state-of-the-art deep learning model as well as a major commericial software system.

7.2 Related Works

While much work has been done on the task of table localization, fewer works attempt to parse table structure.

The ICDAR 2013 Table Competition [4] compared many systems for table structure recognition, including the current state-of-the-art method by Nurminen [9]. This method performs edge detection on the rendered PDF image (filtering out edges due to text) to find junctions and rectangular areas. Then it follows a series of heuristic steps to assign text to rows, identifying likely columns and headers, revising rows, merging columns, find spanning cells, etc. Shigarov et al. later proposed a simpler set of heuristics based on bottom-up merging of PDF components [12]. The TEXUS system [10] performs simple table structure parsing by considering each text chunk to be a cell.

The DeepDeSRT [11] learning model formulates table structure extraction as pixel labeling tasks to segment row objects and column objects. It finetunes the FCN-2s model [8] that has been pretrained on the PASCAL VOC semantic segmentation challenge [2] for

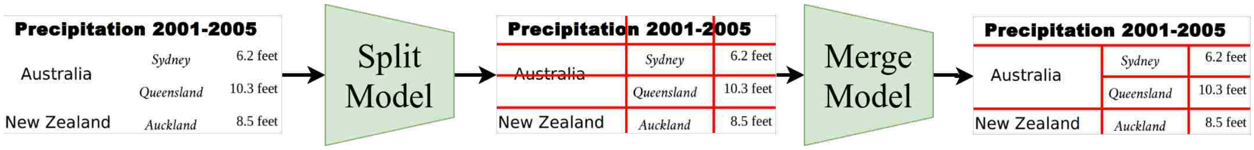


Figure 7.1: Overview of the proposed models. First the Split model predicts the grid of the table. Then the merge model combines grid elements to recover spanning cells.

natural scenes. While it only requires an image as input, it does not perform as well as heuristic methods on PDFs. Clinchat et al. compare two learning models on scanned images of seventeeth century handwritten tables [1]. They focus on this narrow problem and use template matching to recover columns and convert the images into "PDF-like" documents. To recover table rows, they use either a Conditional Random Field (CRF) or their proposed Edge Convolutional Networks, which extends Graph Convolutional Networks [14].

7.3 Deep Model

Our proposed table structure extraction method relies on two deep learning models that are applied in sequence (see Fig. 7.1). We call these models the Split model and the Merge model. The Split model takes an input image of a well cropped table and produces the grid structure of the table in the form of row and column separators that span the entire image. Because some tables contain spanning cells, we apply the Merge model to the grid output of the Split model to merge together adjacent grid elements to recover the spanning cells.

7.3.1 Split Model

The Split model takes as input an image of any dimension $H \times W$ and produces two 1-D output signals: $r \in [0,1]^H$ and $c \in [0,1]^W$. The output signals r and c represent the probability that each row (column) of pixels is part of a logical table row (column) separator region. The Split model is composed of 3 sub-networks:

1. Shared Fully Convolutional Network (SFCN)

2. Row Projection Network (RPN)

3. Column Projection Network (CPN)

The SFCN is composed of 3 convolution layers with 7x7 kernels, with the last layer performing a dilated convolution [15] with a dilation factor of 2. Each layer produces 18 feature maps uses a ReLU activation function.

Dilated convolutions, like pooling, increase the receptive field of the network, but unlike pooling, they preserve the spatial resolution of the input. Preserving the input spatial resolution is important in table structure extraction because many column and row separators are only a few pixels wide. In [11], better results were obtained when resizing the initial input to make the separator regions bigger. It is also crucial to have a large receptive field because determining the locations of row and column separators may require global context. For example, text that is consistently left-justified to the same position is indicative of a column separator.

The output of the SFCN is fed as input to both the RPN and CPN. The output of the RPN is r, the probability that each row of pixels is part of a row separator region. Similarly, the output of the CPN is c. Because the RPN and CPN have identical structure, except for whether projection and pooling operations are over rows or columns of pixels, we arbitrary choose to focus our explanation on the RPN.

The RPN is composed of 5 blocks chained together. Fig. 7.2 shows the operations performed by a single block. First, the input is (in parallel) fed to 3 convolution layers that have dilation factors of 2/3/4 to produce 6 feature maps each. The output of each dilated convolution is concatenated to obtain 18 feature maps. Using multiple dilation factors allows the RPN to learn multi-scale features and increase its receptive field while still examining local evidence.

Next, the RPN performs 1x2 max pooling. This decreases the width of the feature maps, but maintains the height, so that the output signal r is of size H. Only the first three blocks perform max pooling to make sure that the width is not downsampled too much.

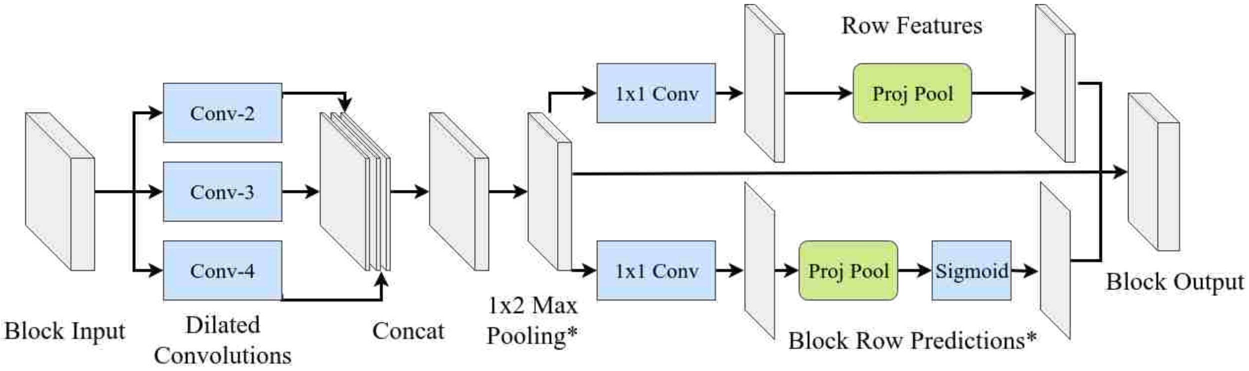


Figure 7.2: A single block of the Row Projection Network (RPN), which is a sub-network of the Split model. Components marked with * indicate that they are not included in all 5 blocks that compose the RPN. Blocks in the Column Projection Network (CPN) use 2x1 Max Pooling.

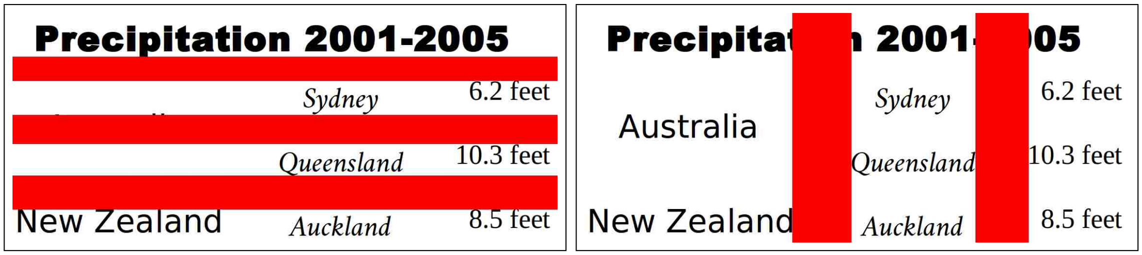
Afterwards, the RPN computes row features (top branch of Fig. 7.2) through a 1x1 convolution operation followed by *projection pooling*. Projection pooling is inspired by the projection profile operation used to find gaps of whitespace in classical layout analysis. With projectin pooling, we maintain the spatial size of the input (instead of collapsing to 1D as in projection profiles), and simply replace each value in the input with its row average. Specifically,

$$\hat{F}_{ij} = \frac{1}{W} \sum_{j'=1}^{W} F_{ij'} \tag{7.1}$$

where i indexes over rows in feature map F, and $1 \leq j \leq W$ indexes over columns of F. We call $\hat{F}$ the row projection pooling of F, and apply this operation independently on each feature map, as is typical of pooling operations. Pooling in this fashion allows information to propagate across the entire width of the image, which may be more than 1000 pixels. These row features are concatenated to the output of the max pooling operation, so that each pixel has both local and row-global features. Note that the CPN performs *column* projection pooling, which similarly is

$$\hat{F}_{ij} = \frac{1}{H} \sum_{i'=1}^{H} F_{i'j} \tag{7.2}$$

The bottom branch of Fig. 7.2 shows how some blocks produce row predictions. A 1x1 convolution produces a single output map, over which we perform projection pooling.



(a) Row

(b) Col

Figure 7.3: Example 1D GT row and column signals for training the Split model. The 1D signals are replicated to 2D and super imposed on the input image for viewing purposes.

We then apply a sigmoid function to produce probabilities. Since each row of pixels contains a single unique probability, we can take a vertical slice to obtain a 1D signal of probabilities, r^n, where n indicates the block index. To make the intermediate prediction r^n available to block $n + 1$, we also concatenate the probabilities (in 2D) to the output of the block.

In our implementation, only the last 3 blocks produce outputs, i.e. r^3, r^4, r^5. During training, we apply a loss to all three predictions, but after training, we only use the last prediction, r^5 for inference. This iterative prediction procedure allows the model to make a prediction and then refine the prediction. Such techniques have been successfully applied in previous work on structured keypoint detection tasks in natural scenes (e.g. human pose [13]).

Training

The SFCN, RPN, and CPN sub-networks are jointly trained in a typical supervised fashion on table images at 150 DPI. We assume that the images are cropped to only include the table's tabular contents, excluding e.g. table titles, captions, and footnotes.

Each table has annotated ground truth (GT) 1D signals r^* and c^*. The GT is designed to maximize the size of the separator regions without intersecting any non-spanning cell content, as shown in Fig. 7.3. This is contrary to the traditional notion of cell separators, which for many tables are thin lines that are only a few pixels thick. Predicting small regions is more difficult than large regions, and in the case of unlined tables, the exact location of a thin separator is ambiguous. The GT separator regions may intersect cells with content that span multiple rows or columns. The goal of the Split model is to recover the basic grid of the table, and spanning cells are later handled by the Merge model.

The loss function is the average element-wise binary cross entropy between the block predictions and the GT signals:

$$L(r, r^*) = \frac{1}{H} \sum_{i=1}^{H} r_i^* \log r_i + (1 - r_i^*) \log(1 - r_i) \tag{7.3}$$

To prevent overfitting, we modify equation 7.3 to clip the per-element loss to 0 when $|r_i^* - r_i| < 0.1$. The total loss is a weighted sum of individual output losses:

$$L_{tot} = L(r^5, r^*) + \lambda_4 L(r^4, r^*) + \lambda_3 L(r^3, r^*) + L(c^5, c^*) + \lambda_4 L(c^4, c^*) + \lambda_3 L(c^3, c^*) \tag{7.4}$$

where we set $\lambda_4 = 0.25$ and $\lambda_3 = 0.1$.

We train the model from random initialization with the ADAM optimizer [7] for approximtately 10^6 weight updates with a batch size of 1. We use an initial learning rate of 0.00075 and decay it by a factor of 0.75 every 80K updates.

Inference

Once we have predicted r, we need to infer at which pixel positions the row separators occur. For simplicity, the discussion focuses on r, but the same procedure applies to c to obtain column separators. To do so, we partition the image into rows and row separator regions by performing a graphcut segmentation [5] over r. Then we choose the row pixel positions corresponding to the midpoints of each inferred separator region.

To create the graph for segmenting r, we have H nodes arranged in a linear chain, where each node is connected to its two neighbors (except the two nodes at either end) with edges of uniform weight $\lambda_{gc} = 0.75$. Node i is connected to the source node (for separator region) with an edge weight of r_i and to the sink node (for row regions) with an edge weight of $1 - r_i$.

7.3.2 Merge Model

The Merge model takes the basic grid structure produced by the Split model and identifies which grid elements need to be merged to recover cells that span multiple rows or columns.

The input is the grid structure rendered as a 2D image concatenated as an extra channel to the original table image. If the grid structure is composed of M rows and N columns, then the model outputs two matrices:

1. D - probability of up-down merges (size $(M-1) \times N$)
2. R - probability of left-right merges (size $M \times (N-1)$)

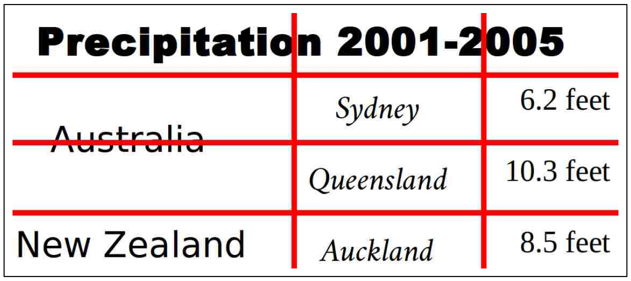
D_{ij} is the probability of merging cell (i, j) with cell $(i+1, j)$ and R_{ij} is the probability of merging cells (i, j) and $(i, j+1)$. In our formulation, all of these probabilities are independent, i.e. a single grid element may be merged in multiple directions.

The Merge model's architecture is similar to the Split model. There is a set of 4 shared convolutional layers (no dilation), where 2x2 average pooling occurs after layers 2 and 4. Afterwards, the model has 4 branches, where each branch predicts an $M \times N$ matrix of probabilities that cells should merge in a particular direction, i.e. up, down, left, or right. Call these matrices u, d, l, r. We then compute D and R as

$$D_{ij} = \frac{1}{2} u_{i+1,j} d_{ij} + \frac{1}{4}(u_{i+1,j} + d_{ij}) \tag{7.5}$$

$$R_{ij} = \frac{1}{2} l_{i,j+1} r_{ij} + \frac{1}{4}(l_{i,j+1} + r_{ij}) \tag{7.6}$$

We only predict that a pair of cells should be merged if there is consistency between the individual branch outputs. While our independence assumption suggests that we multiply the two individual probabilities together in Eqs. 7.5, 7.6, it would introduce optimization difficulties when both probabilities are near 0.



(a) Image with Grid Structure

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

(b) GT D

$$\begin{matrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

(c) GT R

Figure 7.4: Example Merge model GT for the given structure predicted by the Split model.

Each branch is composed of 3 blocks that are similar to the Split model block shown in Fig. 7.2. The differences are that the parallel convolution layers use 1/2/3 dilation factors, no max pooling is performed, and projection pooling is replaced with *grid pooling*. In grid pooling, every pixel location replaces its value with the average of all pixels within its grid element:

$$\hat{F}_{ij} = \frac{1}{|\Omega(i,j)|} \sum_{i',j' \in \Omega(i,j)} F_{i'j'} \tag{7.7}$$

where $\Omega(i,j)$ returns the set of coordinates for all pixels within the grid element that contains (i,j). Thus after grid pooling, all pixels inside the same grid element share the same value, which allows information to propagate within each cell. To produce the u, d, l, or r matrix for a given branch, we average the predictions in each grid element and arrange them in an $M \times N$ matrix. Like the Split model, the Merge model also performs iterative output refinement, where blocks 2 and 3 produce output predictions.

Training

Because the Split and Merge models are intended to be used in sequence, we train the Merge model using the grid structures produced by the Split model. To construct the GT D and R matrices (see Fig. 7.4), we

1. Iterate over all spanning cells in the table
2. Identify which grid elements intersect the GT bounding box of the spanning cell

Category	Method	F-measure	Recall	Precision	PDF Input	Image Input
Proposed Models	Split	86.79	86.64	86.93		✓
	Split-PDF	91.63	91.26	92.00	✓	✓
	Split-PDF + Merge-PDF	90.89	90.44	91.36	✓	✓
	Split-PDF + Merge-PDF + Heuristics	91.95	91.46	92.45	✓	✓
	Split + Heuristics	93.00	92.24	93.78	✓	✓
	Split-PDF + Heuristics	**95.26**	**94.64**	**95.89**	✓	✓
Previously Reported Results	Nurminen [9]	94.60	94.09	95.12	✓	✓
	TEXUS [10]	82.59	84.23	81.02	✓	
	Shigarov [12] C_1	91.50	91.21	91.80	✓	
	Shigarov [12] C_2	93.64	92.33	94.99	✓	
	DeepDeSRT [11]	91.44†	87.36†	95.93†		✓

† result not directly comparable due to evaluation on a random subset of 34 tables.

Table 7.1: ICDAR 2013 Table Dataset Results.

3. Set the probability of merge for each of these cells to 1 for the appropriate directions

As in the Split model, the loss function for each output is the average (clipped) element-wise binary cross entropy (Eq. 7.4). The total loss is then

$$L_{tot} = L(D^3, D^*) + \lambda_2 L(D^2, D^*) + L(R^3, R^*) + \lambda_2 L(R^2, R^*) \tag{7.8}$$

Because spanning cells only occur in 15% of tables in the private dataset used to train our models, we subsample this set so that 50% of the training set for the Merge model has at least one pair of cells that require merging. The training hyperparameters are similar to those of the Split model.

Inference is performed by thresholding D and R at 0.5 probability. We then infer additional merges to ensure that the resulting table structure has only rectangular cells, e.g., no cells in an L-shape.

7.4 Experimental Results

We tested our models on two datasets. The first is the ICDAR 2013 Table Competition standard benchmark [4], where we achieve state-of-the-art results. This dataset is composed of 156 tables extracted from PDFs downloaded from government websites. Only tables with unambiguous structure were included in this dataset. The entire set of PDFs is considered the test data (i.e. no provided training data).

The second dataset is a private collection of 93,000 tables taken from web-scraped PDFs that have been manually annotated with structure information. These tables exhibit a larger variety of complexity, visual appearance, and structure than the ICDAR 2013 dataset. For example, it has a larger proportion of unlined tables, partially lined tables, and tables with non-header multi-line cells. It also contains tables with poorly aligned columns, raster images inside of cells, and multiple sets of column headers.

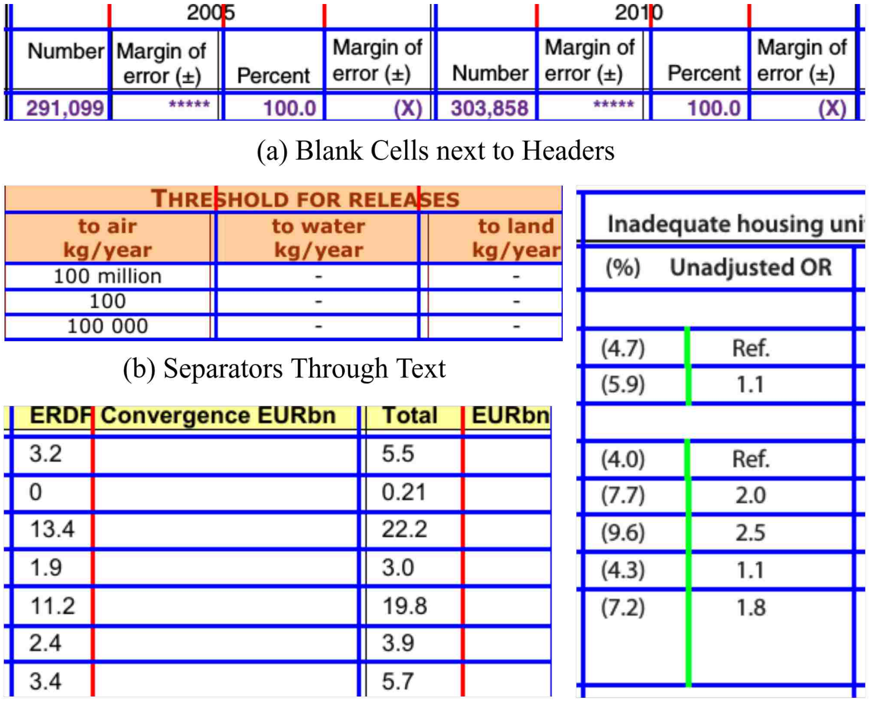
We randomly split this dataset into 83K/5K/5K for training, validation, and testing. Because tables from the same PDF exhibit some similarity, we ensured that all tables from a single PDF were allocated to a single data partition.

The results on the ICDAR 2013 dataset are from models trained on the private collection. We attempted to validate that our improved performance comes from a superior model rather than a larger training set. We did so by reimplementing the DeepDeSRT model [11] and training on the same data as our proposed model.

We also experimented with giving the models additional input channels. If the input table images are from PDFs, then we can extract if pixels are text, vector art (e.g. lines), or raster images by examining drawing command metadata. In addition to the entire grayscale image, we can concatenate additional input channels that contain 1) only text 2) only vector art 3) only raster images. This additional information slightly improves network performance.

7.4.1 ICDAR 2013

Table 7.1 shows the results on the ICDAR 2013 Table Competition dataset (task 2). Our methods with the *-PDF* suffix indicate that the additional PDF input channels were used. The evaluation metric for this dataset is the F-measure over detected *adjacency relationships* [3]. Roughly speaking, this measures the percentage of correctly detected pairs of adjacent cells, where both cells are segmented correctly and identified as neighbors.



(a) Blank Cells next to Headers

(b) Separators Through Text

(c) Join Columns

(d) Split Columns

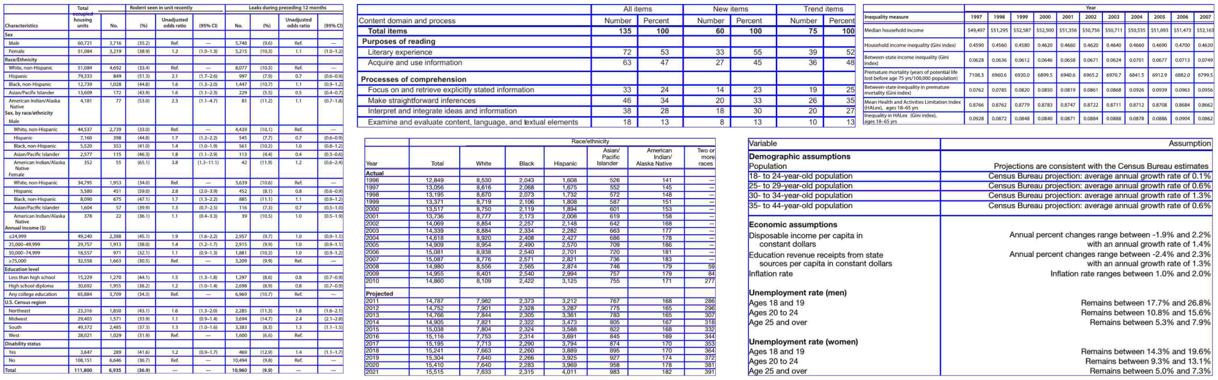
Figure 7.5: Example heuristic post-processing on ICDAR2013 table segments. Removed separators are shown in red, added separators in green, and unmodified predicted separators in blue.

For this dataset, the Merge model failed to provide adequate post-processing for the output of the Split model. Therefore, we also experimented with some simple heuristics to split and join cells. These heuristics include

- Split Columns that have a consistent whitespace gap between vertically aligned text
- Join cells where the predicted separator passes through text
- Join adjacent columns when most non-header adjacent cell pairs have at most 1 cell with text (e.g. merge a content column with a blank column).
- In the first row, merge text cells with any adjacent blank cells.

Some example tables that are fixed by the heuristics are shown in Fig. 7.5.

While the Split model performs well in identifying the underlying table grid, it sometimes makes easily correctable mistakes and it cannot by itself handle spanning cells.



	Total occupied housing units	Rodent seen in unit recently				Leaks during preceding 12 months			
Characteristics		No.	(%)	Unadjusted odds ratio	(95% CI)	No.	(%)	Unadjusted odds ratio	(95% CI)
Sex									
Male	60,721	3,716	(35.2)	Ref.	—	5,748	(9.6)	Ref.	—
Female	51,084	3,219	(38.9)	1.2	(1.0–1.3)	5,215	(10.3)	1.1	(1.0–1.2)
Race/Ethnicity									
White, non-Hispanic	51,084	4,692	(33.4)	Ref.	—	8,077	(10.3)	Ref.	—
Hispanic	79,333	849	(51.3)	2.1	(1.7–2.6)	997	(7.9)	0.7	(0.6–0.9)
Black, non-Hispanic	12,739	1,028	(44.8)	1.6	(1.3–2.0)	1,447	(10.7)	1.1	(0.9–1.2)
Asian/Pacific Islander	13,609	172	(43.9)	1.6	(1.1–2.3)	229	(5.5)	0.5	(0.4–0.7)
American Indian/Alaska Native	4,181	77	(53.0)	2.3	(1.1–4.7)	81	(11.2)	1.1	(0.7–1.8)
Sex, by race/ethnicity									
Male									
White, non-Hispanic	44,537	2,739	(33.0)	Ref.	—	4,439	(10.1)	Ref.	—
Hispanic	7,160	398	(44.8)	1.7	(1.2–2.2)	545	(7.7)	0.7	(0.6–0.9)
Black, non-Hispanic	5,520	353	(41.0)	1.4	(1.0–1.9)	561	(10.3)	1.0	(0.8–1.2)
Asian/Pacific Islander	2,577	115	(46.3)	1.8	(1.1–2.9)	113	(4.4)	0.4	(0.3–0.6)
American Indian/Alaska Native	352	55	(65.1)	3.8	(1.3–11.1)	42	(11.9)	1.2	(0.6–2.4)
Female									
White, non-Hispanic	34,795	1,953	(34.0)	Ref.	—	3,639	(10.6)	Ref.	—
Hispanic	5,580	451	(59.0)	2.8	(2.0–3.9)	452	(8.1)	0.8	(0.6–0.9)
Black, non-Hispanic	8,090	675	(47.1)	1.7	(1.3–2.2)	885	(11.1)	1.1	(0.9–1.2)
Asian/Pacific Islander	1,604	57	(39.9)	1.3	(0.7–2.5)	116	(7.3)	0.7	(0.5–1.0)
American Indian/Alaska Native	378	22	(36.1)	1.1	(0.4–3.3)	39	(10.5)	1.0	(0.5–1.9)
Annual income ($)									
≤24,999	49,240	2,388	(45.1)	1.9	(1.6–2.2)	2,957	(9.7)	1.0	(0.9–1.1)
25,000–49,999	29,757	1,913	(38.0)	1.4	(1.2–1.7)	2,915	(9.9)	1.0	(0.9–1.1)
50,000–74,999	18,557	971	(32.1)	1.1	(0.9–1.3)	1,881	(10.3)	1.0	(0.9–1.2)
≥75,000	32,558	1,663	(30.5)	Ref.	—	3,209	(9.9)	Ref.	—
Education level									
Less than high school	15,229	1,270	(44.1)	1.5	(1.3–1.8)	1,297	(8.6)	0.8	(0.7–0.9)
High school diploma	30,692	1,955	(38.2)	1.2	(1.0–1.4)	2,698	(8.9)	0.8	(0.7–0.9)
Any college education	65,884	3,709	(34.3)	Ref.	—	6,969	(10.7)	Ref.	—
U.S. Census region									
Northeast	23,316	1,850	(43.1)	1.6	(1.3–2.0)	2,285	(11.3)	1.8	(1.6–2.1)
Midwest	29,403	1,571	(33.9)	1.1	(0.9–1.4)	3,694	(14.7)	2.4	(2.1–2.8)
South	49,372	2,485	(37.3)	1.3	(1.0–1.6)	3,383	(8.3)	1.3	(1.1–1.5)
West	28,021	1,029	(31.9)	Ref.	—	1,600	(6.6)	Ref.	
Disability status									
Yes	3,647	289	(41.6)	1.2	(0.9–1.7)	469	(12.9)	1.4	(1.1–1.7)
No	108,151	6,646	(36.7)	Ref.	—	10,494	(9.8)	Ref.	—
Total	111,800	6,935	(36.9)	—	—	10,960	(9.9)	—	—

	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	**135**	**100**	**60**	**100**	**75**	**100**
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

	Year										
Inequality measure	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007
Median household income	$49,497	$51,295	$52,587	$52,500	$51,356	$50,756	$50,711	$50,535	$51,093	$51,473	$52,163
Household income inequality (Gini index)	0.4590	0.4560	0.4580	0.4620	0.4660	0.4620	0.4640	0.4660	0.4690	0.4700	0.4630
Between-state income inequality (Gini index)	0.0628	0.0636	0.0612	0.0646	0.0658	0.0671	0.0624	0.0701	0.0677	0.0713	0.0749
Premature mortality (years of potential life lost before age 75 yrs/100,000 population)	7108.3	6960.6	6920.0	6899.5	6940.6	6965.2	6970.7	6841.5	6912.9	6882.0	6799.5
Between-state inequality in premature mortality (Gini index)	0.0762	0.0785	0.0820	0.0850	0.0819	0.0861	0.0868	0.0926	0.0939	0.0963	0.0956
Mean Health and Activities Limitation Index (HALex), ages 18–65 yrs	0.8766	0.8762	0.8779	0.8783	0.8747	0.8722	0.8711	0.8712	0.8708	0.8684	0.8662
Inequality in HALex (Gini index), ages 18–65 yrs	0.0928	0.0872	0.0848	0.0840	0.0871	0.0884	0.0888	0.0878	0.0886	0.0904	0.0862

	Race/ethnicity						
Year	Total	White	Black	Hispanic	Asian/ Pacific Islander	American Indian/ Alaska Native	Two or more races
Actual							
1996	12,849	8,530	2,043	1,608	526	141	—
1997	13,056	8,616	2,068	1,675	552	145	—
1998	13,195	8,670	2,073	1,732	572	148	—
1999	13,371	8,719	2,106	1,808	587	151	—
2000	13,517	8,750	2,119	1,894	601	153	—
2001	13,736	8,777	2,173	2,008	619	158	—
2002	14,069	8,854	2,257	2,148	642	168	—
2003	14,339	8,884	2,334	2,282	663	177	—
2004	14,618	8,920	2,408	2,427	686	178	—
2005	14,909	8,954	2,490	2,570	709	186	—
2006	15,081	8,938	2,540	2,701	720	181	—
2007	15,087	8,776	2,571	2,821	736	183	—
2008	14,980	8,556	2,565	2,874	746	179	59
2009	14,955	8,401	2,540	2,994	757	179	84
2010	14,860	8,109	2,422	3,125	755	171	277
Projected							
2011	14,787	7,982	2,373	3,212	767	168	286
2012	14,752	7,901	2,328	3,287	775	165	296
2013	14,766	7,844	2,305	3,361	783	165	307
2014	14,905	7,821	2,322	3,473	805	167	318
2015	15,038	7,804	2,324	3,588	822	168	332
2016	15,116	7,753	2,314	3,691	845	169	344
2017	15,195	7,713	2,290	3,794	874	170	353
2018	15,241	7,663	2,260	3,889	895	170	364
2019	15,304	7,640	2,266	3,925	927	174	372
2020	15,410	7,640	2,283	3,969	958	178	381
2021	15,515	7,633	2,315	4,011	983	182	391

Variable	Assumption
Demographic assumptions Population	Projections are consistent with the Census Bureau estimates
18- to 24-year-old population	Census Bureau projection: average annual growth rate of 0.1%
25- to 29-year-old population	Census Bureau projection: average annual growth rate of 0.6%
30- to 34-year-old population	Census Bureau projection: average annual growth rate of 1.3%
35- to 44-year-old population	Census Bureau projection: average annual growth rate of 0.6%
Economic assumptions Disposable income per capita in constant dollars Education revenue receipts from state sources per capita in constant dollars Inflation rate **Unemployment rate (men)** Ages 18 and 19 Ages 20 to 24 Age 25 and over **Unemployment rate (women)** Ages 18 and 19 Ages 20 to 24 Age 25 and over	Annual percent changes range between -1.9% and 2.2% with an annual growth rate of 1.4% Annual percent changes range between -2.4% and 2.3% with an annual growth rate of 1.3% Inflation rate ranges between 1.0% and 2.0% Remains between 17.7% and 26.8% Remains between 10.8% and 15.6% Remains between 5.3% and 7.9% Remains between 14.3% and 19.6% Remains between 9.3% and 13.1% Remains between 5.0% and 7.3%

Figure 7.6: Example predictions for unlined tables, including one failure case

When combined with simple heuristics to handle these cases, it achieves an F-measure of 95.26% compared to the previous best result of 94.60%. The merge model fails to generalize to the ICDAR 2013 dataset, but as shown in Table 7.2, does improve performance on the private collection. Figure 7.6 shows some example predictions by *Split-PDF + Heuristics* for unlined tables, which are harder to recognize than lined tables.

There is a large performance difference from when PDF information (text, path, image channels) is given as input to the Split model and when it is not. Because the difference is not as large on the private collection (Table 7.2), the utility of the PDF information depends on the dataset being considered. The ICDAR tables are primarily lined, have larger headers, and may have distinct visual appearances compared to the training dataset. Thus, the additional PDF information may help more in the unfamiliar domain as the text and path elements are explicitly given rather than visually inferred by the model.

We reimplemented the DeepDeSRT model for table structure extraction [11], but trained on the same private data as our proposed model. However, we were unable to obtain reasonable performance (e.g. above 70% FM) even after exploring a variety of post processing thresholds and training hyperparameters. In [11], they report an FM of 91.44%, but over a random subset of 34 tables, so a direct comparison wouldn't be feasible. Additionally, not all post processing details (e.g. thresholds) were specified in [11] and the training data we used was different. Together, these reasons could explain our observed performance difference, but

Category	Method	Table Accuracy	F-measure	Recall	Precision	PDF Input	Image Input
Proposed Models	Split	80.79	95.46	**95.87**	95.06		✓
	Split + Merge	83.09	95.78	95.50	96.07		✓
	Split-PDF	81.07	95.37	95.64	95.10	✓	✓
	Split-PDF + Merge-PDF	**85.65**	**95.92**	95.59	**96.27**	✓	✓
Our Implementation	DeepDeSRT	51.76	76.85	75.32	78.44		✓
	DeepDeSRT-PDF	53.48	78.44	77.03	79.91	✓	✓
Commerical	Acrobat	61.11	82.62	82.66	82.59	✓	

Table 7.2: Private Collection Results.

as we cannot be sure that we have a faithful reimplementation, we omit the exact performance numbers to avoid direct comparison.

7.4.2 Private Collection

On this dataset, we evaluate methods using precision and recall over correctly detected cells. We also report accuracy as the percentage of tables with perfect precision and recall. A predicted bounding box (BB) is a correct prediction if it entirely contains only a single ground truth cell content BB. In particular, predicted BBs that intersect mulitple GT BBs or that do not completely contain any GT BB are marked as false positives. Unmatched GT BBs are marked as false negatives. Because blank cells were not manually annotated, we exclude predicted BBs that do not intersect any GT BBs. This way, methods are not penalized if they correctly predict unmarked blank cells.

Table 7.2 shows the results on the test split of 5000 tables. Following [4], reported precision and recall are computed per-table and then averaged. We were unable to find any official implementations of prior works, so for comparison, we use the commerical software system Acrobat Pro DC and our reimplementation of the DeepDeSRT model [11].

All variants of our proposed models outperform the two baselines across all measures by a significant amount. We also see that the merge model significantly improves table accuracy and that using PDF information as input does bring an improvement, but more marginally than with the ICDAR 2013 data. This suggests that this method may be effective as well on scanned documents, where only the captured image is available for input.

7.5 Conclusion

We have proposed a pair of deep learning models that together split a table image into the basic grid of cells, and then merge cells together to recover cells that span mulitple rows and columns. The key insight of the models is to pool information over large regions of the table image such as entire rows/columns of pixels or previously predicted cell regions. When evaluating the split model on the ICDAR 2013 Table Competition dataset, we achieve state-of-the-art performance. We also show that PDF information, such as if page elements are text/path/image, can be encoded as input to the deep network and improve performance. However, if such information is not available (e.g. scanned documents), the model can take just the grayscale image as input. Lastly, we have shown that the Merge model is effective using a private collection of tables extracted from the web.

References

[1] Stéphane Clinchant, Hervé Déjean, Jean-Luc Meunier, Eva Maria Lang, and Florian Kleber. Comparing machine learning approaches for table recognition in historical register books. In *Document Analysis Systems (DAS)*, pages 133–138. IEEE, 2018.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, Springer, 2010.

[3] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. A methodology for evaluating algorithms for table understanding in PDF documents. In *Symposium on Document Engineering (DocEng)*, pages 45–48. ACM, 2012.

[4] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. ICDAR 2013 table competition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1449–1453. IEEE, 2013.

[5] Dorothy M. Greig, Bruce T. Porteous, and Allan H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, JSTOR, 1989.

[6] Jianying Hu, Ramanujan Kashi, Daniel Lopresti, George Nagy, and Gordon Wilfong. Why table ground-truthing is hard. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 129–133. IEEE, 2001.

[7] Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *International Conference on Learning Representation (ICLR)*, ArXiv, 2014.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 3431–3440. IEEE, 2015.

[9] Anssi Nurminen. Algorithmic extraction of data in tables in PDF documents. Master's thesis, Tampere University of Technology, 2013.

[10] Roya Rastan, Hye-Young Paik, and John Shepherd. TEXUS: A task-based approach for table extraction and understanding. In *Symposium on Document Engineering (DocEng)*, pages 25–34. ACM, 2015.

[11] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. DeepDeSRT: Deep learning for detection and structure recognition of tables in document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1162–1167. IEEE, 2017.

[12] Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. Configurable table structure recognition in untagged PDF documents. In *Symposium on Document Engineering (DocEng)*, pages 119–122. ACM, 2016.

[13] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732. IEEE, 2016.

[14] Qiongkai Xu, Qing Wang, Chenchen Xu, and Lizhen Qu. Collective vertex classification using recursive neural network. *arXiv preprint arXiv:1701.06751*, 2017.

[15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Part III

Recognition Tasks

Chapters 8-10 concern three common recognition tasks in document image analysis. Chapter 8 is about whole image classification, such as determining if an image is a letter, scientific article, or email. It is a large scale empirical study that examines what factors influence the accuracy of a CNN classifier in this domain. Recognizing the optical font to produce an image of text is the subject of Chapter 9. It also examines the handwritten equivalent of fonts: medieval scribal scripts. Handwriting recognition is one of the most common tasks in document image analysis and is covered in Chapter 10. This chapter examines leveraging training data in a high resource language, such as English, to train a recognizer for another language with a similar alphabet (e.g. French, Italian) that lacks annotated images. Finally, Chapter 11 provides some concluding remarks.

Chapter 8

Analysis of Convolutional Neural Networks for Document Image Classification

Abstract

Convolutional Neural Networks (CNNs) are state-of-the-art models for document image classification tasks. However, many of these approaches rely on parameters and architectures designed for classifying natural images, which differ from document images. We question whether this is appropriate and conduct a large empirical study to find what aspects of CNNs most affect performance on document images. Among other results, we exceed the state-of-the-art on the RVL-CDIP dataset by using shear transform data augmentation and an architecture designed for a larger input image. Additionally, we analyze the learned features and find evidence that CNNs trained on RVL-CDIP learn region-specific layout features.

Tensmeyer, Chris, and Tony Martinez. "Analysis of Convolutional Neural Networks for Document Image Classification." *International Conference on Document Analysis and Recognition (ICDAR).* Vol. 1, pp. 388-393. IEEE, 2017.

8.1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have been proposed for document image classification and have enjoyed great success. These impressive results include:

- 100% accuracy on NIST tax forms dataset [3] using only one training sample per class [8].
- 89.8% accuracy on the RVL-CDIP genre classification dataset. The Bag of Words baseline was 49.3% [4].
- 96.5% accuracy in fine-grained classification of identity documents compared to 92.7% accuracy using Histogram of Oriented Gradients (HOG) features [16].

However, many such applications of CNNs to whole document image classification [1, 4, 16] use a CNN pre-trained on the ImageNet dataset of natural images [13] as a starting point for features. Through this transfer learning approach is effective due to the generality of CNN features [14], there exist large domain differences between natural images and document images. For example, in ImageNet, the object of interest can appear in any region of the image in a variety of 3D poses. In contrast, many document images are 2D entities that occupy the whole image. Due to such domain differences, we question whether the same architectures and techniques that are effective for natural images are also optimal for document images.

To answer this query, we conducted a large empirical study utilizing two large document image datasets. To our knowledge, we are the first to conduct such a study for the domain of document images. We examine many factors that contribute to CNN classification accuracy, which can be broadly categorized as data preprocessing and network architecture. All CNNs in this work are randomly initialized and not pretrained on natural images.

For data preprocessing, we examine what representation(s) of the image (e.g. binary, RGB, HSV), are most effective as input to the network. We also test 10 different types of label-preserving image transformations (e.g. crop, rotation, shear) to artificially expand the training data. While cropping is typically applied in CNNs trained on ImageNet, we find that shear transforms yield best performance for document image tasks. While not all document images are the same aspect ratio (AR), CNNs typically only accept inputs of a fixed size (e.g. 227x227) and hence a fixed AR. We investigate this issue and find that CNNs trained with stochastic shearing are not adversely affected by AR warping of input images.

For network architectures, we examine factors such as network depth, width, and input size under various amounts of training data. Critically, we achieve 90.8% accuracy on RVL-CDIP *without pretraining on natural images* by using a larger input size This surpasses the previous best result of 89.8% [4] on this dataset. By incorporating multi-scale images into training and inference, we reach 91.03%. We also examine non-linear network operations in the domain of document images.

Lastly, we analyze what intermediate features CNNs learn from document image tasks. Though CNNs are often considered black-box models, individual neurons can be characterized by their maximal-exciting inputs. We find evidence that CNNs learn a wide variety of region-specific layout features. Several intermediate neurons fire on page elements of specifics shapes and types (e.g. typed text, handwritten text, graphics).

8.2 Related Work

CNNs have been used in document image analysis for two decades for tasks such as character recognition [11, 15], but only more recently were applied to large image classification tasks. In 2012, Krizhevsky et al. showed the effectiveness of CNNs in large image classification by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a large margin [9]. Since then, all top entries of ILSVRC have been based on CNNs, rather than on handcrafted features, with the 2015 winner surpassing human level performance by utilizing residual connections and batch normalization in a CNN with 152 layers [6].

In 2014, one of the first applications of CNNs to whole document image classification used a 4-layer CNN to classify tax forms and the Small Tobacco datasets [8]. This CNN acheived 65.37% accuracy on Small Tobacco compared to ∼42% by the previous state-of-the-art HVP-RF classifier [10], which uses SURF local descriptors.

The following year, two works explored transferring the parameters of a CNN learned on ILSVRC to document classification tasks. The DeepDocClassifier system [1] retrained the top classification layer from scratch while finetuning the other layers. They report an accuracy of 77.3% on Small Tobacco. Harley et al. [4] also transferred parameters from ILSVRC, but also introduced a larger dataset, called RVL-CDIP, that can be used to train CNNs from scratch.They also found that a single holistic CNN outperformed an ensemble of region-specific CNNs on RVL-CDIP.

8.3 Convolutional Neural Networks

The CNNs we consider in this work are models that map input images $x \in \mathbb{R}^{H\times W\times D}$ into the probability vectors $y \in \mathbb{R}^C$, where D is the input image depth (e.g. 3 for RGB) and C is the number of classes. Each layer, of the CNN performs an affine transformation with learnable parameters followed by non-linear operation(s):

$$x_\ell = g_\ell(W_\ell \star x_{\ell-1} + b_\ell) \tag{8.1}$$

where $1 \leq \ell \leq L$ is the layer index, x_0 is the input image, W_ℓ, b_ℓ are learnable parameters, $\star$ is either 2D convolution (with multi-channel kernels) for convolution layers or matrix multiplication for fully connected layers, and g_ℓ is a layer specific non-linearity, composed of $\text{ReLU}(x) = \max(0, x)$, and optionally max-pooling, local response normalization, or dropout [9]. The output of the last layer, x_L, is input to a softmax function, which outputs a probability vector over the target classes. For more details, consult [9].

Many of our experiments are based on the standard AlexNet architecture [9] (5 conv layers, 3 fully connected layers), but without the original sparse connections in the convolutional layers. We also test many architectural variations, which are noted in the relevant experiments.

8.4 Training Details

8.4.1 Datasets

For this work, we utilize 2 datasets. The first is the publicly avaialable RVL-CDIP dataset [4] which is composed of 400,000 grayscale images split into 320,000/40,000/40,000 train/val/test sets. They are scanned office documents with 16 conceptual categories such as Letter, Memo, Email, Form.

The second dataset, denoted *ANDOC*, is composed of genealogical records sampled from 974 collections owned by `Ancestry.com`. The target class of each image is its collection of origin (e.g. 1940 US Census, Pennsylvania death certificates 1880-1940). In total, there are 880,000 images partitioned into a randomized 800,000/40,000/40,000 train/val/test split. 481,000 of these images are color, while the rest are grayscale.

For data preprocessing, pixel intensities are scaled to $[0, 1]$ and then the channel mean is subtracted.

8.4.2 Training Hyperparameters

We empirically determined good training schemes for each dataset. For RVL-CDIP, CNNs are trained with Stochastic Gradient Descent (SGD) with mini-batches of 32 for 500,000 weight updates. The initial learning rate (LR) is 0.003 and is decayed by a factor of 10 every 150,000 mini-batches. For ANDOC, we used SGD with mini-batches of 128 for 250,000 weight updates. The initial LR was 0.005 and is decayed by a factor of 10 every 100,000 mini-batches. We believe the larger mini-batch does better for ANDOC because it has more output classes.

All networks are trained on the training split and progress is monitored on the validation set. The set of network parameters that performed best on the validation set is then evaluated on the test images. In this work, we use test set accuracy as our evaluation metric.

CNNs require days to train on high-end GPUs, so training many CNNs for statistical significance testing is often too time-consuming. For this reason, CNN literature almost always reports numbers from only a best single trained model. Thus we typically report average performance of 1-2 CNNs. However, in Section 8.5.1, we trained 10 CNNs for each set of hyperparameters in order to measure the variance in model accuracy, which we estimate to be $\sigma \approx 0.1$ for RVL-CDIP and $\sigma \approx 0.05$ for ANDOC.

8.5 Preprocessing Experiments

8.5.1 Image Representation

Here we examine whether the best input representation is RGB, HSV, grayscale (G), binary (B), dense-SURF (S) [2], or a combination. For example, when combining RGB and G, we treat the input as a 4-channel image where channels 1-3 are RGB and channel 4 is G. For D-SURF inputs, we compute SURF descriptors at fixed orientation on a 227x227 grid on the *originally sized* images. This results in a 64-dimensional descriptor for each grid point, which we treat as an image with 64 channels. Binary images are computed with Otsu's method [12].

Figure 8.1 shows average accuracy of 10 CNNs. Overall, combining RGB or G with S leads to the best accuracy. Though S by itself does not perform well, we believe it combines well with RGB/G because the SURF descriptors are computed using the originally sized images. As we later show, using larger input images lead to significant gains in performance. Using B leads to worst performance and augmenting RGB/G with B leads to lower performance. In ANDOC, the HSV colorspace performed equally as well as RGB, though combining both leads to a marginal increase.

For simplicity, the rest of the experiments use G input for RVL-CDIP and RGB input for ANDOC. Additional experiments (results not shown) suggest that with optimal data

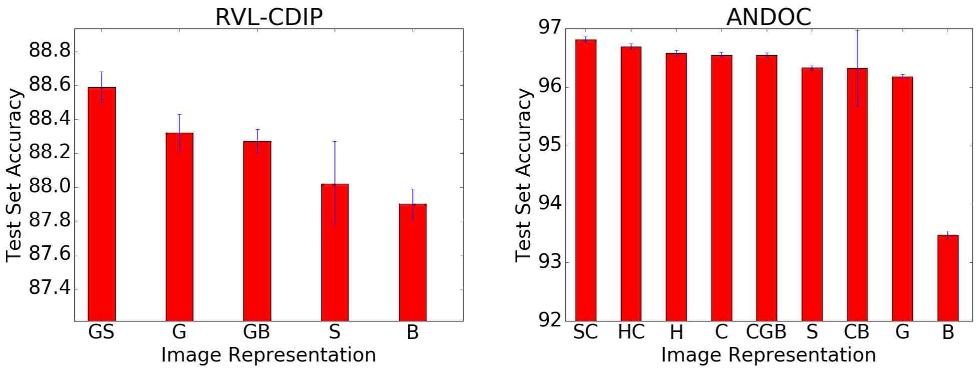


Figure 8.1: Mean accuracy of 10 CNNs as we vary the image representation. Error bars indicate the standard deviation of 10 trials. G=Grayscale, C=RGB, H=HSV, S=dense-SURF, B=Binary.

augmentation (Section 8.5.2), augmenting RGB/G with S does not lead to significant gains in performance.

8.5.2 Data Augmentation

It is common practice to stochastically transform each input during SGD training to artificially enlarge the training set to improve performance [6, 9]. We experimented with 10 types of transformations (e.g. crop, blur, rotation) and 2-4 parameter settings for each transformation type for a total of 38 CNNs trained with different data augmentation. We report results for the best CNNs per transform type in Table 8.1. Following [17], we also report multiple-view test performance where the overall CNN prediction is the average of the predictions made on 10 transforms of each test image. This increases computation for prediction by 10x, but can increase accuracy [9], which makes it appropriate for some applications. The transforms used at test time are the same type as those used during training and include the untransformed image. While some of these transforms are commonly used, shear, perspective, and elastic transforms have not been.

Shear transforms perform best for single-view testing and are comparable to the best multiple-view transforms. When images are sheared (either horizontally or vertically), the relative locations of layout components are perturbed, but unlike rotation or perspective

Transform	RVL-CDIP 1x Test	RVL-CDIP 10x Test	ANDOC 1x Test	ANDOC 10x Test
None	88.30	88.30	96.54	96.54
Color Jitter	88.36	88.34	96.43	96.41
Crop	88.83	**89.31**	96.50	96.57
Elastic [15]	87.94	88.12	96.27	96.36
Gaussian Blur	87.50	87.46	96.31	96.32
Gaussian Noise	88.19	88.18	96.57	96.47
Mirror	88.51	88.93	96.69	**96.79**
Perspective	88.63	88.45	96.59	96.64
Rotation	88.74	77.46	96.37	96.36
Salt/Pepper Noise	88.03	87.97	96.39	96.45
Shear	**89.33**	84.88	**96.75**	96.59

Table 8.1: Best performance for 10 types of data augmentation

transforms, either the horizontal or vertical structure is preserved. We believe that compared to other transform types, shearing best models the types of intra-class variations in document datasets. The CNNs in the remaining experiments were trained using shearing with $\theta \in [-10°, 10°]$.

We then attempted to combine the two or three best performing types of transformations. However, this did not improve performance over using a single transform for either single or multi-view testing (results omitted for brevity).

8.5.3 Aspect Ratio

One potential drawback of the standard CNN is fixed spatial input dimensions (e.g. 227x227). This means all inputs must have the same aspect ratio (AR), though the original images may have various ARs. For example, some document images are in portrait or landscape orientation or are printed on different paper sizes. Figure 8.2 shows histograms of AR's in both datasets, which is more varied for ANDOC. These images of various AR must be transformed to match the same expected CNN input dimensions. A common technique resizes the image to the input size without maintaining AR. We refer to this as *inconsistent AR warping.*

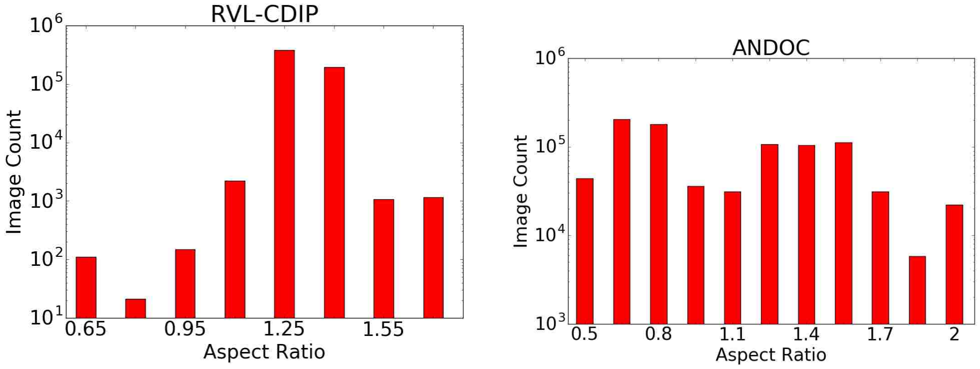


Figure 8.2: Histogram of Aspect Ratios. Note y-axes are log scale.

In this experiment, we compare some alternatives to inconsistent AR warping in the context of document images. One method pads images to the correct AR before resizing, however this wastes some precious input content. Another method resizes (preserving AR) the image until the smallest dimension is the correct size. Next, crops of the correct size are taken as input images. At test time, predictions are averaged over 3 crops so the CNN sees all parts of the image.

A third way is to modify the CNN architecture to accept variable AR inputs. While the convolution layers can operate on any input size, the fully connected layers expect a fixed sized vector input. One way around the input size requirement is to replace the fixed 2x2 pooling regions of the last convolution layer with a Spatial Pyramid Pooling (SPP) operation [5]. This way the size of the pooling regions vary with the input size and always yield the same sized vector output. Inspired by [10], we experimented with pooling regions arranged in horizontal and vertical partitions (HVP), but it did not outperform SPP pooling, likely due to the small input size (13x13) at this layer.

We experimented with two different input sizes, 227x227 and 384x384. The 384x384 CNN has the same overall structure as AlexNet, but is wider and has a greater amount of downsampling. The variable AR inputs for the SPP-CNN were resized to have the same or fewer number of pixels as the fixed sized inputs. In Table 8.2 we report the average test accuracy of 2 CNNs on each type of input. Overall, no alternative outperformed inconsistent

Method	RVL-CDIP 227x227	RVL-CDIP 384x384	ANDOC 227x227	ANDOC 384x384
Warped	89.25	90.84	96.73	97.14
Padded	89.02	90.89	96.59	97.03
Padded + SPP	88.85	**90.94**	96.59	97.12
Warped + SPP	**89.31**	90.92	**96.77**	**97.21**
SPP (variable AR)	88.51	88.71	96.20	97.15
Variable AR + Crop†	88.68	89.94	96.57	97.05

† Predictions averaged over 3 crops.

Table 8.2: Comparison of ways to address aspect ratio warping

AR warping even with the SPP which accepts inputs of various AR. Our hypothesis is that stochastically shearing input images (Section 8.5.2) makes the CNN more invariant to inconsistent AR warping.

8.6 Network Architecture Experiments

8.6.1 Depth

The computational time of a CNN for both training and inference is influenced by the number of layers (depth) and the number of neurons in each layer (width).

To modify depth, we removed or added convolution layers to the AlexNet architecture, always keeping the first two layers and maintaining the same downsampling. For shallower nets, we removed layers in this order: conv3, conv4, conv5. For deeper nets, we inserted layers identical to conv3 between conv3 and conv4, where no pooling occurs. We also ran these experiments where the size of the training set was reduced to 50%, 10%, 1% for RVL-CDIP and 50%, 10%, 2% for ANDOC.

Figure 8.3 gives the results for varying network depth. Here we observe an overall decay in performance as depth increases, even for large amounts of training data. This decay especially pronounced when less less than 10% of the data is used (<32K or <79K instances), as increasing depth past 2 layers leads to sharp decreases in accuracy. This validates the

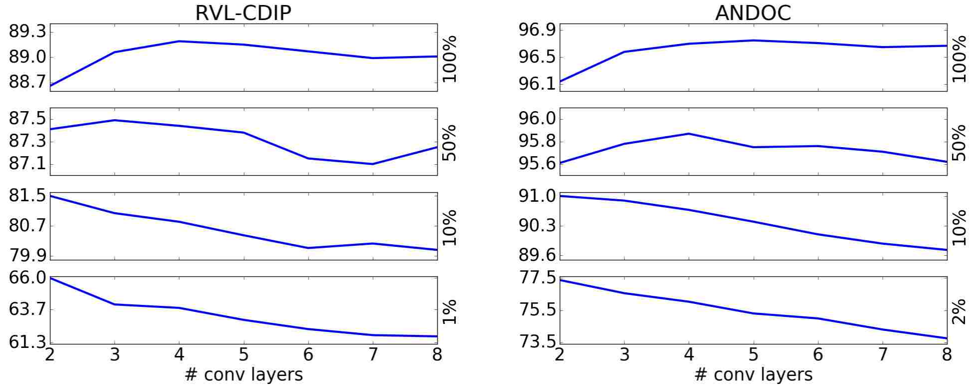


Figure 8.3: Accuracy vs network depth. Each sub-graph is for a different training set size, which is indicated to the right of the graph.

architectural choice of a 2-conv layer network of [8]. These results suggest that network depth should adapt to the size of the training dataset.

We also observe the diminishing returns of additional data for CNN approaches, with only a 1-2% decrease in performance with 50% of the training data. This suggests that gathering more data (e.g. millions of training examples) will only marginally improve CNN performance and more data efficient methods should be sought for.

8.6.2 Width

To modify width, we multiply the number of neurons in each layer by a constant factor. We also experiment with changing just the width of the convolution layers or just the width of the fully connected layers. Results for RVL-CDIP (ANDOC results were similar) are presented in Table 8.3.

With the full training set, network performance saturates at 100% of the AlexNet width. For smaller datasets, smaller widths are optimal. We also see that reducing the width of only the convolutional layers leads to lower performance than just reducing the width of the fully connected layers.

Width	100% Data	50% Data	10% Data	1% Data
10%	82.80	82.70	77.95	62.12
25%	87.38	86.23	80.53	**63.71**
50%	88.75	87.26	**80.68**	63.16
75%	88.99	**87.39**	80.59	63.42
100%	89.18	87.15	80.52	63.48
125%	89.22	87.16	80.31	62.79
150%	89.20	87.22	80.46	62.98
200%	**89.23**	87.09	80.06	62.37
fc-50%	**89.17**	**87.20**	**80.79**	**63.68**
conv-50%	88.78	87.09	80.61	63.24

Table 8.3: Accuracy of various width CNNs on RVL-CDIP with different amounts of training data

8.6.3 Input Size

Standard CNN architectures [6, 9, 17] accept inputs that are 224x224, but to our knowledge, this design choice has not been thoroughly explored before. At this resolution, large text is generally legible, but smaller text is not. We empirically test square input sizes $\{n \times n | n = 32, 64, 100, 150, 227, 256, 320, 384, 512\}$. Of necessity, we modify the architecture of the CNN for each input size with the following principles.

1. Same number of layers
2. Increase kernel size and network width with input size
3. Spatial output size of final convolution layer is 6x6

An exhaustive grid search over possible architectures was not possible with our computational resources and we acknowledge that our chosen architectures may not be optimal. Figure 8.4 shows a distinct trend with larger inputs leading to increased performance. In fact, increasing the input size to 384x384 for RVL-CDIP yields an accuracy of 90.8%, which exceeds the previous published best result of 89.9% [4].

Given that input size significantly impacts performance, we went further to evaluate multi-scale training and testing of CNNs on document images. We did this by using the architecture of the largest input size and replacing the last fixed-sized pooling regions of

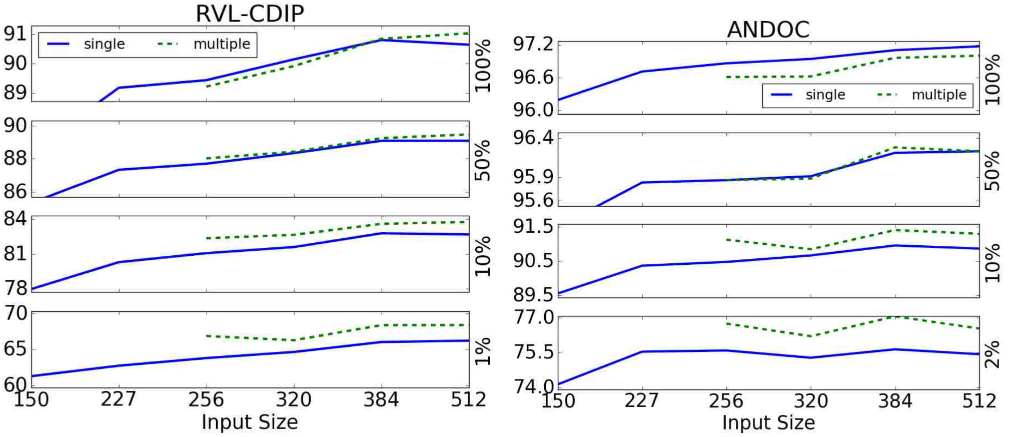


Figure 8.4: Accuracy vs input size. Solid blue lines are for CNNs trained with a single input size. Dashed green lines are for CNNs trained with variable sized input images. The x-axis location corresponds to the largest input size used to train the CNN. For example, the performance of the CNNs trained with images of size 256-384 is plotted at x=384.

the CNN with SPP pooling regions [5] (see Section 8.5.3). During training, all images are randomly resized within a prespecified range (spanning 3 sizes in Figure 8.4). At test time, we average predictions across 3 sizes and report the results in Figure 8.4. This may be considered another type of data augmentation (Section 8.5.2), but only works for architectures that can process multiple sizes. This further improves performance on RVL-CDIP to 91.03% for a CNN trained on images of size 320-512 and predictions averaged across sizes 320,384,512. Smaller datasets also have higher relative improvement using multi-scale training and testing.

8.6.4 Non-Linearities

Here we test whether LRN or Dropout non-linear components contribute to the overall CNN performance. We also examine a relative new technique termed Batch Normalization (BN) that has been shown to both improve performance and increase training speed [7]. BN works by first linearly scaling and shifting each neuron's activations to have zero mean and unit variance. These statistics are calculated from the activations caused by just the images in each mini-batch. This is then followed by a learnable shifting and scaling of the activation values so that the mean and variance of the neuron's activation values is learned. We insert BN after each convolution or matrix multiplication and before ReLU.

Non-linearity	RVL-CDIP	ANDOC
LRN + Dropout + BN	**89.25**	97.45
LRN + Dropout	89.21	96.72
LRN + BN	89.03	**97.57**
LRN	87.63	95.98
Dropout	89.17	96.72

Table 8.4: Accuracy based on Non-linearities

Results are shown in Table 8.4. For ANDOC, BN improves performance by a large margin and can replace dropout. This is likely due to the visual variety of the documents (due to diverse set of classes) in ANDOC. In contrast, Dropout is better than BN in RVL-CDIP, likely due to the visual uniformity of office-style documents. We also observe that LRN also does not increase performance for either dataset.

8.7 Analyzing What Is Learned

One critique of CNNs is that they lack interpretation, making it difficult to know what features they use for classification decisions. We analyzed a CNN trained on RVL-CDIP by examining the top-9 input patches that excite neurons and the *deconv* visualization of each of these patches [19]. The deconv visualization runs the network backwards from the neuron to the input in the context of some input image or patch. It highlights the salient parts of the image that led to the neuron firing. We found this technique critical for deciphering patterns recognized by neurons in later layers because the effective input size (i.e. receptive field) is very large. For this analysis, we utilized the visualization tool proposed in [18].

In our analysis we manually categorized the first 100 neurons in each convolutional layer (see Table 8.5). Although the category of some neurons is subjective, our labeling captures the trends in the features learned by each layer. As expected, the first two layers detect simple elements which are gradually abstracted into complex detectors for text and arrangement of elements. While our category labels were the same for all layers, the complexity of each category increased for deeper layers. It is noteworthy that almost all neurons were

Category	conv1	conv2	conv3	conv4	conv5
Edge	10	2	0	0	0
Parallel lines	46	9	10	9	2
Checker	5	1	2	2	0
Stroke(s)	10	10	2	0	0
Shape Pattern(s)	15	32	24	14	16
Corner	5	25	13	5	3
Small Text	3	4	8	23	42
Large Text	0	16	20	23	12
Column/Margin	0	0	11	5	9
Table Cells	1	0	2	0	2
Graphic	0	1	4	4	0
Handwritten Text	0	0	0	0	3
Scanner Noise	0	0	0	0	3
Ambiguous	0	0	2	15	8

Table 8.5: Categorization of 100 neurons for each convolution layer.

interpretable, with the majority of the *Ambiguous* category being neurons that fired on two distinct types of features. This was not the case for the 4-layer network of [8], whose first layer filters appear random.

Neurons in conv5 tended to find particular configurations of text, such as those shown in Figure 8.5. One interpretation is that the CNN is performing a loose form of layout analysis as an intermediary step to classification. The fully connected layers can then reason about the spatial correlation of these elements to form a classification decision. We also observed that by conv5, the CNN has learned to distinguish between handwritten text and type-set text, though no explicit information about the type of text was provided to the CNN.

We also examined the spatial specificity of features by averaging the intermediate output images for each filter across all images of RVL-CDIP. As seen in Figure 8.6, conv1 filter responses are generally not confined to any portion of the image. However, the other layers exhibit many filters that only respond to certain regions, which is consistent with the hypothesis that CNNs learn layout features.

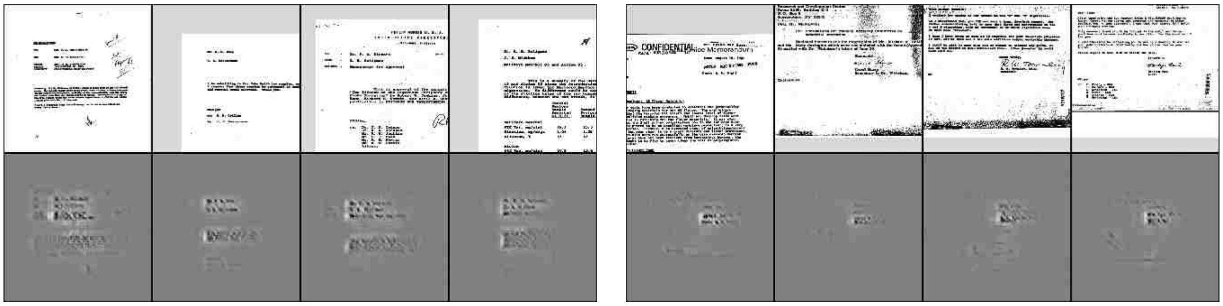
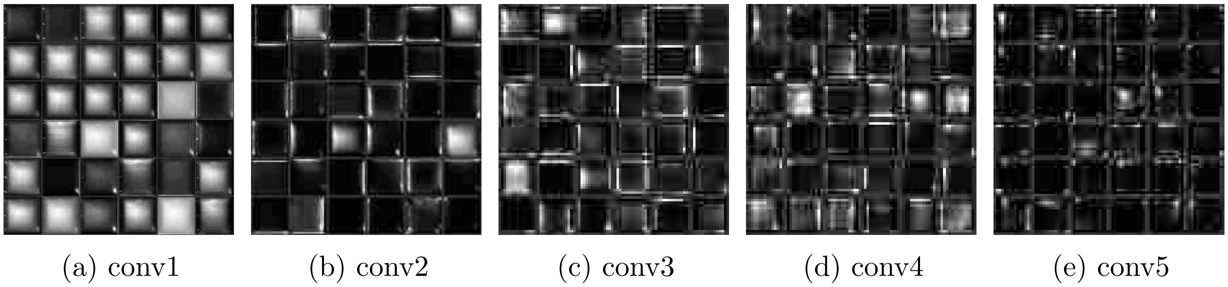


Figure 8.5: Top 4 activating patches and deconv visualizations for 2 text detection neurons in conv5. The neuron detects left justified headers followed by wide paragraphs of text. The bottom neuron detects right justified text below paragraphs of text. Both features are class discriminative (e.g. *Letter* class).



(a) conv1 (b) conv2 (c) conv3 (d) conv4 (e) conv5

Figure 8.6: Average of filter responses for 36 filters for convolution layers. Layers conv2-5 have filters that are specific to spatial regions.

8.8 Conclusion

We examined several factors that influence CNN performance on document images. Overall, applying shear transforms during training and using large input images lead to the biggest gains in performance, acheiving state-of-the-art performance on RVL-CDIP at 90.8% accuracy. Multi-scale training and testing also improve performance, specifically for smaller training sets. As well, BN is a useful alternative to Dropout in datasets that have large visual variety.

We also examined a CNN trained on RVL-CDIP and found evidence that the CNN is learning intermediate layout features. Neurons fire based on type of layout component (graphic, text, handwriting, noise, etc) and tend to fire on specific locations on the image.

References

[1] Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki. Deepdocclassifier: Document classification with deep convolutional neural network. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE, 2015.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, Elsevier, 2008.

[3] D. L. Dimmick, M. D. Garris, and Wilson C. L. NIST structured forms reference set of binary images (SFRS). Online, 1991. URL `http://www.nist.gov/srd/nistsd2.cfm`.

[4] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, IEEE, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML*, volume 37, pages 448–456. JMLR, 2015.

[8] Le Kang, Jayant Kumar, Peng Ye, Yi Li, and David Doermann. Convolutional neural networks for document image classification. In *International Conference on Pattern Recognition (ICPR)*, pages 3168–3172. IEEE, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. MIT Press, 2012.

[10] Jayant Kumar and David Doermann. Unsupervised classification of structurally similar document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1225–1229. IEEE, 2013.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, IEEE, 1998.

[12] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, IEEE, 1979.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, Springer, 2015.

[14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–813. IEEE, 2014.

[15] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962. IEEE, 2003.

[16] Marcel Simon, Erik Rodner, and Joachim Denzler. Fine-grained classification of identity document types with only one example. In *International Conference on Machine Vision Applications (MVA)*, pages 126–129. IEEE, 2015.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[19] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

Chapter 9

Convolutional Neural Networks for Font Classification

Abstract

Classifying pages or text lines into font categories aids transcription because single font Optical Character Recognition (OCR) is generally more accurate than omni-font OCR. We present a simple framework based on Convolutional Neural Networks (CNNs), where a CNN is trained to classify small patches of text into predefined font classes. To classify page or line images, we average the CNN predictions over densely extracted patches. We show that this method achieves state-of-the-art performance on a challenging dataset of 40 Arabic computer fonts with 98.8% line level accuracy. This same method also achieves the highest reported accuracy of 86.6% in predicting paleographic scribal script classes at the page level on medieval Latin manuscripts. Finally, we analyze what features are learned by the CNN on Latin manuscripts and find evidence that the CNN is learning both the defining morphological differences between scribal script classes as well as overfitting to class-correlated nuisance factors. We propose a novel form of data augmentation that improves robustness to text darkness, further increasing classification performance.

Tensmeyer, Chris, Daniel Saunders, and Tony Martinez. "Convolutional Neural Networks for Font Classification." *International Conference on Document Analysis and Recognition (ICDAR).* Vol. 1, pp. 985-990. IEEE, 2017.

9.1 Introduction

Deep Convolutional Neural Networks (CNNs) have been successfully applied to many problems in Document Image Analysis. These areas include whole image classification [1, 4, 7], image preprocessing [12], script identification [16], and character recognition [18]. The success of CNNs has been attributed to their ability to learn features in an end-to-end fashion from large quantities of labeled data.

In this work, we present a simple CNN based framework for classifying page images or text lines into font classes. Handling multiple fonts is a challenge in Optical Character Recognition (OCR), as the OCR system must handle large variations in character appearance due to differences in font. If text lines are labeled with a font class, then a specialist OCR system for that font can potentially achieve higher recognition rates than an OCR system trained on many fonts [2, 17].

In this framework, a CNN is trained to classify small image patches into font classes. At prediction time, we densely extract patches from the test image and average font predictions over individual patch predictions. Although this method is simple, we achieve 98.8% text line accuracy on the King Fahd University Arabic Font Database (KAFD) for 40 type faces in 4 styles and 10 different sizes [9]. The best previous result is 96.1% on a subset of 20 type faces [9]. We also demonstrate state-of-the-art performance with 86.6% accuracy on the Classification of Latin Medival Manuscripts (CLaMM) dataset, where the highest previously reported accuracy is 83.9% [3].

In addition to showing that CNNs perform well at font classification tasks, we perform an in-depth analysis of the features learned by the CNN. Though CNNs are black box models, we can gain an understanding of what features are used for classification by measuring output responses as we vary characteristics of the input images. Such an analysis can demonstrate whether the CNN is overfitting to nuisance factors of the collection of documents it was trained on.

For example, the CLaMM dataset contains 12 scribal script classes defined by expert paleographers that are handwriting styles that differ in character allographs and morphological shape [3]. However, we find that CNNs trained on CLaMM are sensitive to how dark the text is. This is undesirable because the CNN may be applied to novel document collections that have a different bias w.r.t. text darkness. We provide a solution to this problem using a new form of data augmentation, which also improves performance on CLaMM. We also find that CNNs can be sensitive to other factors such as inter-line spacing and presence of non-textual content.

9.2 Related Works

We review the literature for two tasks: font classification and analyzing what features a CNN has learned. Though the classes in CLaMM are referred to as *script classes*, we feel that this task is more akin to font classification than what is traditionally described as script classification. Traditional script classification deals with distinguishing different writing systems or character sets (e.g. Chinese vs Latin vs Arabic), while script classes in CLaMM are all Latin script. However, in keeping with the terminology introduced in [3], we use the term *script* to refer to a class category in CLaMM. We also use the term *font class* to refer to typefaces (e.g. Arial) rather than combinations of typeface, size, and style (e.g. bold).

9.2.1 Font Classification

Zramdini and Ingold presented a font recognition system based on the statistics of connected components, achieving 97.35% accuracy over English text lines for 10 font classes [23]. Zhu et al. posed font recognition as texture identification and used Gabor Filters to achieve 99.1% accuracy on text blocks for both English and Chinese text [22]. Fractal Dimension features were introduced in [10] for Arabic font classification and resulted in 98% accuracy for 10 font classes. Luqman et al. used log-Gabor filter features extracted at multiple scales and orientations to obtain 96.1% accuracy on 20 fonts in the large scale KAFD dataset.

More recently, deep learning techniques based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been proposed for font classification. Tao et al. used a combination of CNN and 2D RNN models to classify single Chinese characters into 7 font classes with 97.77% accuracy [20]. Pengcheng et al. classified handwritten Chinese characters into 5 calligraphy classes with 95% accuracy using deep features extracted from a CNN pretrained on natural images [13]. Classifying calligraphy classes is similar to classifying script types in CLaMM, though CLaMM uses Latin script classes with page-level ground truth.

9.2.2 Analysis of Learned Features

There are a variety of techniques used to examine features learned by CNNs, developed primarily in the context of natural images. In [19], canonical class images are obtained by gradient descent over input pixels to maximally excite output class neurons. Similarly, class sensitivity maps are visualized by computing via backpropagation the first partial derivative of the target output class w.r.t. all input pixels. Zeiler and Fergus proposed a *deconv* visualization approach where hidden representations are projected back into the input space [21]. Additionally, they visualize intermediate neurons by finding the top-N maximally exciting input patches for each neuron. For quantitative analysis, Zeiler and Fergus measured how CNN outputs change in response to certain transformations of the input image (e.g. rotation, translation) [21]. We perform a similar analysis, specific to the domain of document images, to see how sensitive CNNs are to noise factors in CLaMM.

9.3 Methods

In this work, we classify large document images into script or font classes using CNNs. Because inputting entire high resolution images to a CNN is computationally slow, requires large GPU memory, and requires more training data, we resort to a patch classification scheme. We train a CNN to classify individual 227x227 patches into font/script classes.

To obtain a classification for a large test image, we densely extract overlapping 227x227 patches on a regular square grid with 100 pixels between patch centers. The CNN produces a probability distribution over the classes for each patch, which distributions are uniformly averaged to obtain the final classification.

For training data, we extract 256x256 patches at a stride of 42 pixels from the training images and label patches with the class of the image it was taken from. During training, a random 227x227 crops from these patches are inputted to the CNN. Some training images are set aside as a validation set, which is used to select the best performing model.

In this work, we compare two CNN architectures. The first is the AlexNet architecture composed of 5 convolution layers followed by 3 fully connected layers [8]. Each layer takes as input the output of the previous layer:

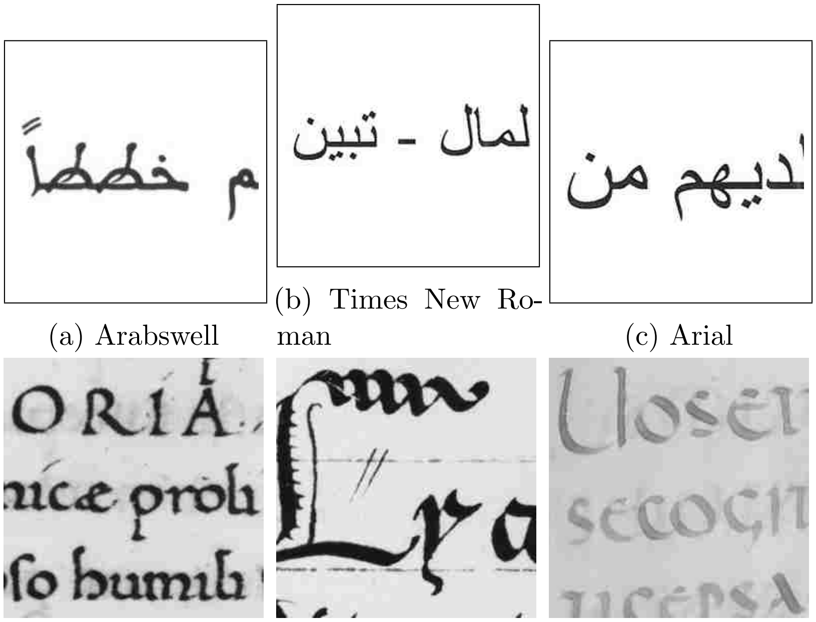
$$x_\ell = F_\ell(x_{\ell-1}, \theta_\ell) \tag{9.1}$$

where x_ℓ is the input to the ℓ^{th} layer, F_ℓ is a function to be learned parameterized by θ_ℓ. The F_ℓ performs either a convolution or a matrix multiplication followed by $\text{ReLU}(z) = \max(z, 0)$, and sometimes pooling and local response normalization operations [8].

The other architecture is the state-of-the-art ResNet-50 [5], which was used to win the 2015 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Layers in a ResNet learn a *residual function*, where F_ℓ is added to the layer input:

$$x_\ell = F_\ell(x_{\ell-1}, \theta_\ell) + x_{\ell-1} \tag{9.2}$$

Residual learning enables deeper networks to be trained with gradient descent optimization as the gradient no longer vanishes or explodes exponentially with layer depth due to the $x_{\ell-1}$ term in Equation 9.2. For ResNet-50, F_ℓ is composed of multiple convolutions, ReLU, BatchNorm [6], and sometimes pooling operations. See [5] for the exact model specification.



(a) Arabswell (b) Times New Roman (c) Arial

(d) Humanistic (e) Hybrida (f) Uncial

Figure 9.1: Example 256x256 patches from KAFD (top) and CLaMM (bottom). Subcaptions refer to the class label of the patch.

Additionally, we train CNNs at 7 image scales from 30-100%, where 100% is original image size and 50% is downsampled by a factor of 2.. This is done by downsampling training images before extracting 256x256 patches and by downsampling tests images for classification. For smaller image scales, more characters are available per 256x256 training patch, but with less detail. For CLaMM experiments, we ensemble networks by averaging predictions of two CNNs trained from different random initializations. For training details, all CNNs are trained with Stochastic Gradient Descent (SGD) using 0.9 momentum and L2 weight decay of 0.0005. The number of iterations was 350K-650K based on architecture and dataset, with mini-batches of size 40-64.

9.3.1 Datasets and Evaluation

We use two datasets in this work, Classification of Latin Medieval Manuscripts (CLaMM) [3] and the King Fahd University Arabic Font Database (KAFD). Example training patches from each dataset are shown in Figure 9.1.

The CLaMM dataset was introduced in the recent *ICFHR2016 Competition on the Classification of Medieval Handwritings in Latin Script* [3]. It is comprised of 2000 training images and 1000 test images (for task 1) in grayscale format. Each image is approximately 1700x1200 pixels (300 dpi) and is composed of handwritten text, background space, and graphics. The training and test images are annotated with a single *script class*. There are 12 script classes representing different styles of character shapes used by scribes in producing handwritten manuscripts in Europe in the years 500 C.E. to 1600 C.E. For CLaMM, we report model accuracy over the 1000 test images.

KAFD is comprised of 115,068 scanned pages of printed Arabic text in grayscale format divided among 40 font classes (e.g. Arial). Each font class contains pages that differ in font size (8-24 point) and style (regular, bold, italic, bold and italic). Though KAFD is available in four resolutions, we opted to use only 300dpi images for computational reasons. We used the designated train/validation/test split and evaluated model accuracy on both the page and line formats of the dataset. Because line images are less than 256 pixels in height, we pad them with white background to be 256 pixels in height. When extracting patches, we discard patches with minimum value > 100 as these patches likely consist of only background.

9.3.2 Pretraining on Synthetic Data

For tasks with limited data, pretraining networks on similar tasks tends to improve performance [15]. Pretraining is performed by initializing network weights to those learned on the pretraining task, except for the classification layer, which is initialized randomly due to the tasks having different classes.

For CLaMM, we experimented with pretraining on a 27-class synthetic font recognition task designed to mimic some character differences between CLaMM classes. We hope that by pre-conditioning the CNN to examine individual characters, it will more easily discriminate script classes based on their defining morphological differences. 23 font classes were based on

Train Data	Model	Lines		Page-Lines		Pages	
		Patch	Image	Patch	Image	Patch	Image
Lines	AlexNet	95.3	97.1	95.3	98.7	38.5	90.8
	ResNet	**97.9**	**98.8**	**97.9**	**99.2**	58.0	92.7
Pages	AlexNet	86.2	88.4	86.0	91.1	58.2	98.2
	ResNet	91.9	93.2	91.7	94.8	**60.5**	**98.5**

Table 9.1: Accuracy of CNNs on KAFD test data at both the patch and whole image level.

the *Liberation Serif* font and 4 more were based on other fonts. Each class deviates from the basic font by a random combination of the following modifications:

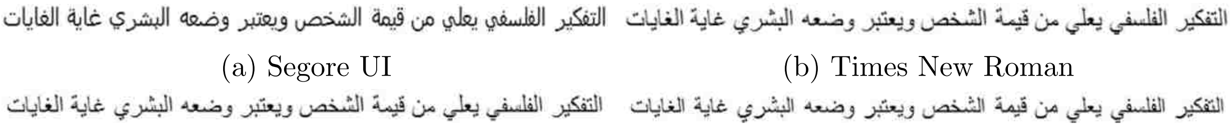
- Only capital glyphs.
- Vertical translation of certain glyphs. Some normally descending characters are shifted to be non-descending, and normally non-descending characters become descenders.
- Substituting glyphs with characters from CLaMM classes. For example, using the Uncial *A*, a single compartment Cursiva *a*, or long *s* glyph instead of modern *s*.

To create training data for a synthetic font class, we rendered random Latin text and added Gaussian noise to foreground characters. We then inserted the noised text onto blank background pages taken from real historical documents using image interpolation. The accuracy on the pretraining task itself ranged from 50-85% depending on CNN architecture and image scale, indicating that the task is non-trivial and that CNNs are able to focus on individual characters to make classification decisions.

9.4 Results

9.4.1 KAFD

On KAFD, we trained 2 ResNets and 2 AlexNet CNNs each on line images and on page images at 100% image scale. For each CNN architecture and type of training data, we selected one model using the provided validation data. Model accuracies for both patches and images are shown in Table 9.1. The *Page-Lines* column shows page level accuracy obtained



التفكير الفلسفي يعلي من قيمة الشخص ويعتبر وضعه البشري غاية الغايات

(a) Segore UI

التفكير الفلسفي يعلي من قيمة الشخص ويعتبر وضعه البشري غاية الغايات

(b) Times New Roman

التفكير الفلسفي يعلي من قيمة الشخص ويعتبر وضعه البشري غاية الغايات

(c) Arial

التفكير الفلسفي يعلي من قيمة الشخص ويعتبر وضعه البشري غاية الغايات

(d) Arabic Transparent

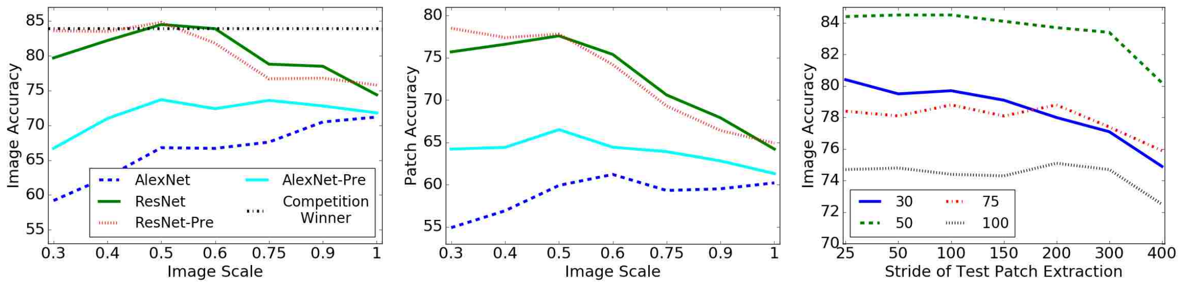
Figure 9.2: KAFD classes that are easily confused, accounting for approximately 70% of misclassifications in all trained models.

by averaging predictions over the pre-segmented line images for each page. For the *Pages* column, test patches are densely extracted from the page image and may contain badly cropped text, leading to lower patch accuracy for this column.

For all types of data, the more powerful Resnet architectures outperforms AlexNet, providing 19-58% error reduction. Training and testing on segmented line images leads to the best page level classification at 99.2% accuracy for ResNet. When training and testing on densely cropped patches, the best accuracy is 98.5%, showing that using segmented data leads to a 47% reduction in error over densely cropped patches. The best accuracy achieved over line images is 98.8%.

To our knowledge, there are no previously published results on all 40 fonts. On subsets of 10 and 20 fonts, Luqman et al. achieved 99.5% and 96.1% accuracy respectively on line images using log-Gabor features extracted at multiple scales and orientations [9]. On the same 20 fonts, a single ResNet model achieves 99.8% accuracy on line images.

The vast majority of misclassifications on all 40 fonts are made by confusing *Times New Roman* with either *Segore UI*, *Arial*, or *Arabic Transparent*. These classes account for at least 70% of errors for patches, line images, and page images. Figure 9.2 shows example text for these 4 easily confused classes. The fonts *Times New Roman*, *Arial*, and *Arabic Transparent* are very similar, and could be grouped together under a single OCR system. However, *Segore UI* is visually distinctive (e.g. larger holes inside characters) and shows that there is room for improvement in the model.



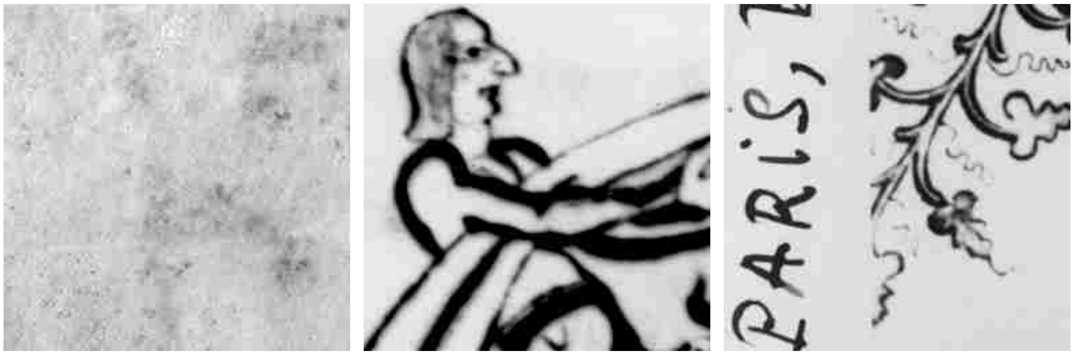
(a) (b) (c)

Figure 9.3: Performance on CLaMM for CNNs trained at difference scales on (a) whole image classification and (b) per-patch classification. *-Pre* indicates that CNNs were first pretrained on synthetic data (Section 9.3.2). A pretrained ResNet trained at 50% image scale performs best and outperforms the previous best result on this task. Subfigure (c) shows the effect of patch extraction stride at test time. Smaller strides are more important for smaller image scales.

9.4.2 CLaMM

For CLaMM, Figure 9.3 shows the performance of ensembles composed of two models, where each ensemble is trained at a single image scale. On this task, we see the importance of using the more powerful ResNet architecture. In general, mid-range image scales perform best, likely because they balance the trade-off between amount of text on each patch and resolution of the text. Notably, using 50% image scale with the ResNet architecture yields 84.5% accuracy, which outperforms the highest reported result of 83.9% achieved by the ICFHR 2016 CLaMM competition winner [3].

Pretraining on synthetic data significantly improves performance for AlexNet models and shows that this is a viable approach to improving CNNs based on domain knowledge. For ResNet, pretraining helps for smaller image scales, though it causes a slight decrease in performance for some larger scales. This is likely because overfitting training data is a bigger problem with smaller scales due to having fewer training patches. Pretraining on the synthetic data at 50% image scale leads to an increase of 0.3% absolute accuracy to reach 84.8% on this task. We note that we created only one set of pretraining data and did not tweak the pretraining classes to improve results, either on test or validation data.



(a) Background (b) Figure (c) Annotation

Figure 9.4: Example of non-textual patches from CLaMM dataset. Though they do not contain text of the target script class, some of these patches are discriminative of the script class of the page.

We also analyzed the stride at which patches are extracted from test images (Figure 9.3c). Larger strides require less computation because fewer patches are evaluated, but also result in lower accuracy. Smaller strides seems to be more important for smaller image scales, likely because fewer patches are extracted from smaller images at any given stride.

9.5 Analyzing Learned Features

In this section, we analyze how sensitive a ResNet model is to noise factors present in the CLaMM dataset. We take two approaches. In the first approach, we modify the training data and evaluate the performance of the ResNet on the task images. In the second, we take the trained ResNet and measure how changing certain characteristics of input patches affects the classification decision for the patch.

9.5.1 Modified Training Data

Because training and testing patches are densely extracted, not all patches contain text that can be used to discriminate the page image's script class. Such patches may contain only background, figures, or institutional annotations (see Figure 9.4). To measure whether these patches postively or negatively affect model performance, we manually annotated[1] each of

[1] These annotations are available for download at `http://axon.cs.byu.edu/clamm`

	Average	Caroline	Cursiva	Half-Uncial	Humanistic	Humanistic	Hybrida	Praegothica	Semihybrida	Semitextualis	Southern	Textualis	Uncial
Training Set						Cursive					Textualis		
CLaMM	74.4	97.7	65.1	84.4	89.0	91.1	37.5	94.0	37.3	82.4	64.6	51.8	100
CLaMM-Filtered	77.4	97.7	65.1	91.1	92.7	92.4	34.1	94.0	49.4	88.2	68.3	58.8	100
CLaMM-Extended	77.7	96.5	58.1	95.6	91.5	93.7	31.8	91.7	55.4	85.3	79.3	56.5	100
CLaMM-Noise	21.5	15.1	9.3	15.6	39.0	26.6	18.2	48.8	15.7	22.1	7.3	20.0	21.8

Table 9.2: Whole image class accuracies on CLaMM test images with ResNet at 100% image scale for modified training sets.

the 2000 training images using the PixLabler Tool [14] and foreground masks computed using Otsu binarization [11].

We created three modified training sets with the annotations:

1. CLaMM-Filtered (12 classes) - Non-textual patches are removed from the training set.
2. CLaMM-Extended (15 classes) - Non-textual patches are reassigned to one of *background*, *figure*, *annotation*. At test time, predictions for these classes are not allowed
3. CLaMM-Noise (13 classes) - All textual patches are removed from the 12 script classes (manually verified), leaving only non-textual patches as examples of the script classes. An additional *Text* class is composed of textual patches drawn from all classes. At test time, predictions for the *Text* class are not allowed.

We trained an ensemble of two ResNets at 100% image scale (test patch stride of 100) on the three modified training sets and on the unmodified training set. We chose 100% image scale so that *CLaMM-Noise* could have a sufficient number of training patches. The per-class and average accuracies are shown in Table 9.2.

Interestingly, we see that either filtering or relabling non-textual patches increases accuracy by 3%, indicating that it is detrimental to label non-textual patches as examples of script classes. Because non-textual patches are distinctly labeled in *CLaMM-Extended*, the ResNet can minimize their impact at test time. This data preprocessing outperforms that of *CLaMM-Filtered*, where the ResNet at test time must classify non-textual patches into one of the 12 script classes. The results on *CLaMM-Noise* demonstrate that some script classes can be discriminated based on non-textual content, as the average accuracy is 21.5%, compared to 8.3% for random predictions. For example, images containing *Praegothica* and

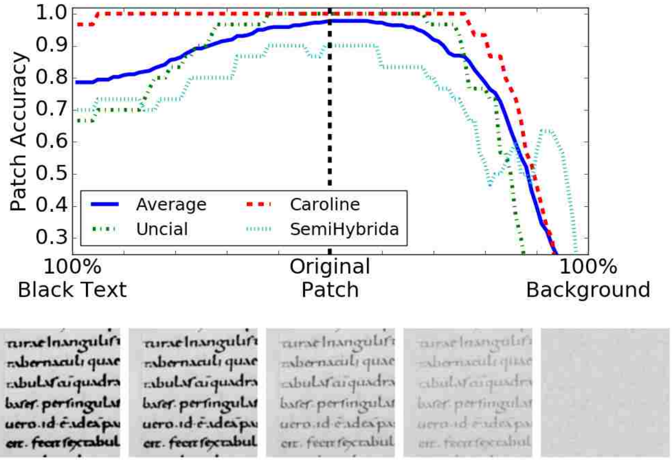


Figure 9.5: Patch Accuracy as a function of text darkness. Patches with varying text darkness below are positioned relative to the x-axis of the graph. Some classes are more or less sensitive to these effects.

Humanistic scripts frequently have large decorated figures, which can be sufficient to classify the images. Some other reasons for performance above random chance include correlation of figures, background intensity, presence of noise (e.g. bleed through) with page level script classes. This shows that CNNs have the potential to overfit to particular characteristics of the collection used to train (and evaluate) the model, so caution should be exercised when applying models to novel collections.

9.5.2 Text Darkness

We also examined learned features by varying input patches and measuring per-patch output accuracy. In preliminary analysis, we found two nuisance factors that influence classification decisions: text darkness and inter-line spacing.

To test the effect of text darkness on patch-accuracy, we extracted 30 patches of text from each class and computed reasonable foreground masks using Otsu binarization [11]. For each patch, we produced a series of 100 patches, where 50 have darker text and 50 have lighter text compared to the original patch. To make text darker, we subtracted constant values from all foreground pixels and clipped values at 0. The value of the constant was linearly varied such that the intensity of all foreground pixels in the darkest image are uniformly

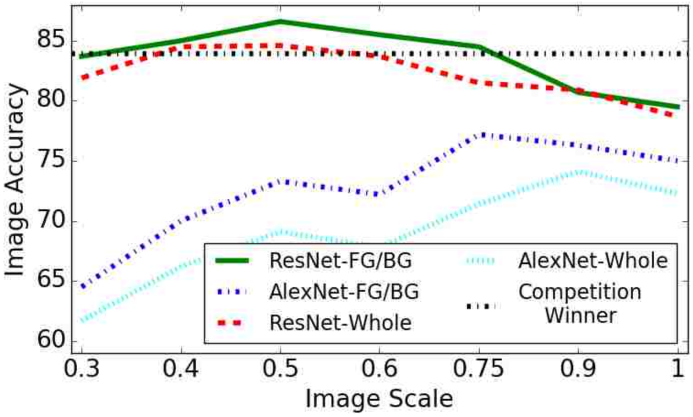


Figure 9.6: Comparing independently varying background and foreground intensity (-*FG/BG*) vs varying whole image intensity (-*Whole*) on CLaMM.

0. To make the text lighter, we produced linear combinations of the original patch and an estimated background image constructed by averaging together sampled background patches. These linear combinations range from 100% original image to 100% background estimate. We chose interpolation over directly lightening text to avoid making any foreground pixels lighter than the surrounding background.

We ordered these 100 modified patches with the darkest as the 1st, the original patch as the 50th, and the lightest patch as the 100th. We classified each modified patch and recorded the average accuracy for each darkness/lightness level for all patches and each class (Figure 9.5). Overall, classification is robust to small changes, but there are sharp decreases for larger changes. For example, the training data for *Uncial* script has uniformly light text, with an average foreground intensity of 130 (other classes have average intensity ∼50). We observe that when *Uncial* text is darkened or lightened, more errors are made because such examples are not present in the training data. However, lightness or darkness of foreground text is not a defining characteristic of the script, but is an artifact of writing instrument, ink composition, and document preservation conditions. The majority of other scripts, such as *SemiHybrida*, exhibit similar sensitivities, while *Caroline* script appears robust to both darkening and lightening of text.

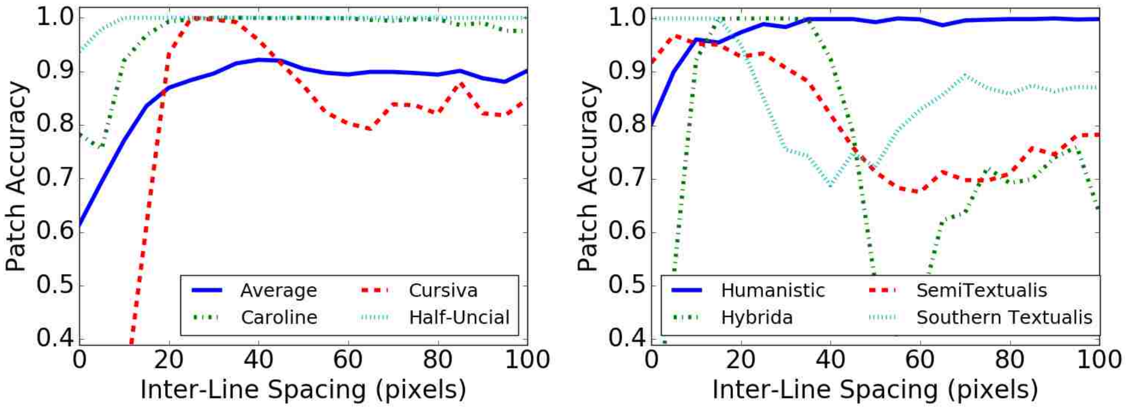


Figure 9.7: Patch accuracy as a function of text line spacing for selected images. Split into two graphs for clarity.

To make CNNs robust to varying text darkness, we apply a novel form of data augmentation. While it is common to randomly brighten or darknen input images on-the-fly during training (e.g. [8]), we independently lighten or darken foreground and background pixels based on masks computed using Otsu binarization [11]. Specifically, each time an image is input to the CNN, we choose either foreground or background with equal probability. Then, we draw a random value from a Gaussian distribution with $\mu = 0, \sigma = 30$ and add that value to the grayscale pixel values of the selected region.

In Figure 9.6 we compare this scheme to simply brightening or darkening the whole image. In general, independently varying foreground and background leads to significant performance gains for all image scales. In particular, we reach a new record at 86.6% accuracy on CLaMM, which exceeds the previous state-of-the-art by an absolute 2.7%.

9.5.3 Line Spacing

We also found that CNNs are sensitive to the inter-line spacing of text for certain classes. For each class, we manually extracted text lines and backgrounds from two images and were able to render those text lines at various spacings on the original backgrounds. We then measured patch-accuracy as a function of text line spacing for each image.

Patch accuracies of selected examples of sensitive images are shown in Figure 9.7. The majority of images examined are most easily classified at line spacing of 20 pixels. For 4 classes, accuracies stayed above 95% for line spacings greater than 20 pixels. For 3 of these classes, sharp drops in accuracy were observed for spacings less than 20 pixels (especially at 0 pixels). Only *Praegothica* images were completely invariant to line spacing. Trends are not necessarily tied to the script class, as we observed that for 5 of 12 classes, the two images examined had drastically different sensitivities to line spacing.

We note that inter-line spacing is not part of the morphological definition of CLaMM classes. Therefore it would be desirable to have predictive models that are not sensitive to line spacings. Though we have not experimentally verified such, we hypothesize that data augmentation where line spacings are stochastically altered (e.g. with seam carving) would give CNNs more invariance to this nuisance factor and potentially make them more accurate for application to novel collections.

Additional experiments (omitted for space) suggested CNNs are sensitive to the line height (i.e. font size) of text lines.

9.6 Conclusion

We have presented a simple patch based classification framework for line image and page image font classification. We have shown that the ResNet architecture in our framework gives state-of-the-art performance by exceeding previously published results in Arabic font classification on the KAFD dataset and in Latin scribal script classification on the CLaMM dataset. We performed an analysis of the sensitivities of ResNet to nuisance factors in the CLaMM dataset, such as non-textual patches, text darkness, and line spacing. In the case of text darkness, we proposed novel data augmentation based on independently varying foreground and background intensities, which leads to improved model robustness and performance.

References

[1] Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki. Deepdocclassifier: Document classification with deep convolutional neural network. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE, 2015.

[2] Henry S Baird and George Nagy. Self-correcting 100-font classifier. In *IS&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology*, pages 106–115. International Society for Optics and Photonics, 1994.

[3] Florence Cloppet, Véronique Eglin, Van Kieu, Dominique Stutzmann, and Nicole Vincent. ICFHR2016 competition on the classification of medieval handwritings in Latin script. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 590–595. IEEE, 2016.

[4] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML*, volume 37, pages 448–456. JMLR, 2015.

[7] Le Kang, Jayant Kumar, Peng Ye, Yi Li, and David Doermann. Convolutional neural networks for document image classification. In *International Conference on Pattern Recognition (ICPR)*, pages 3168–3172. IEEE, 2014.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. MIT Press, 2012.

[9] Hamzah Luqman, Sabri A. Mahmoud, and Sameh Awaida. KAFD Arabic font database. *Pattern Recognition*, 47(6):2231–2240, Elsevier, 2014.

[10] Sami Ben Moussa, Abderrazak Zahour, Abdellatif Benabdelhafid, and Adel M Alimi. New features using fractal multi-dimensions for generalized Arabic font recognition. *Pattern Recognition Letters*, 31(5):361–371, Elsevier, 2010.

[11] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, IEEE, 1979.

[12] J. Pastor-Pellicer, S. España-Boquera, F. Zamora-Martínez, M. Zeshan Afzal, and Maria Jose Castro-Bleda. Insights on the use of convolutional neural networks for document image binarization. In *International Work-Conference on Artificial Neural Networks*, pages 115–126. Springer, 2015.

[13] Gao Pengcheng, Gu Gang, Wu Jiangqin, and Wei Baogang. Chinese calligraphic style representation for recognition. *International Journal on Document Analysis and Recognition*, 20(1):59–68, Springer, 2017.

[14] Eric Saund, Jing Lin, and Prateek Sarkar. Pixlabeler: User interface for pixel-level labeling of elements in document images. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 646–650. IEEE, 2009.

[15] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–813. IEEE, 2014.

[16] Baoguang Shi, Xiang Bai, and Cong Yao. Script identification in the wild via discriminative convolutional neural network. *Pattern Recognition*, 52:448–458, Elsevier, 2016.

[17] Hongwei Shi and Theo Pavlidis. Font recognition and contextual processing for more accurate text recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 39–44. IEEE, 1997.

[18] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962. IEEE, 2003.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[20] Dapeng Tao, Xu Lin, Lianwen Jin, and Xuelong Li. Principal component 2-D long short-term memory for font recognition on single Chinese characters. *Transactions on Cybernetics*, 46(3):756–765, IEEE, 2016.

[21] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[22] Yong Zhu, Tieniu Tan, and Yunhong Wang. Font recognition based on global texture analysis. *Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1192–1200, IEEE, 2001.

[23] Abdelwahab Zramdini and Rolf Ingold. Optical font recognition using typographical features. *Transactions on Pattern Analysis and Machine Intelligence*, 20(8):877–882, IEEE, 1998.

Chapter 10

Language Model Supervision for Handwriting Recognition Model Adaptation

Abstract

Training state-of-the-art offline handwriting recognition (HWR) models requires large labeled datasets, but unfortunately such datasets are not available in all languages and domains due to the high cost of manual labeling. We address this problem by showing how high resource languages can be leveraged to help train models for low resource languages. We propose a transfer learning methodology where we adapt HWR models trained on a source language to a target language that uses the same writing script. This methodology only requires labeled data in the source language, unlabeled data in the target language, and a language model of the target language. The language model is used in a bootstrapping fashion to refine predictions in the target language for use as ground truth in training the model. Using this approach we demonstrate improved transferability among French, English, and Spanish languages using both historical and modern handwriting datasets. In the best case, transferring with the proposed methodology results in character error rates nearly as good as full supervised training.

Tensmeyer, Chris, Curtis Wigington, Brian Davis, Seth Stewart, Tony Martinez, and Bill Barrett. “Language Model Supervision for Handwriting Recognition Model Adaptation.” *International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Vol. 1, pp. 133-138. IEEE, 2018.

10.1 Introduction

State-of-the-art offline handwriting recognition (HWR) models are based on deep Convolutional Neural Networks (CNNs) and Bidirectional Long-Short Term Memory (BLSTM) networks and are trained on large amounts of labeled line images [9]. Obtaining such large annotated training sets is expensive and time consuming, because a person must segment thousands of text lines and manually type transcriptions for the ground truth. However, such a process is often necessary for each language and domain because trained HWR models often fail to generalize sufficiently across domains, languages, and writers that were not observed during training. Eliminating or lessening this requirement is the goal of unsupervised HWR and related approaches.

Prior work has attempted to address the lack of a large labeled training set for low resource languages/domains through several means. Training on synthetic data is an appealing direction because an arbitrary amount of labeled data may be generated with little human effort. In some works, synthetic data is obtained by applying annotation preserving transformations to real data in order to simulate the natural variability in handwriting [19, 21, 23]. However, these methods depend on the availability of sufficiently diverse labeled data, which is not always the case. Other works have modeled the writing process for generating isolated characters using prototypes for Chinese [20] and Korean [12] characters, though it is not clear how such models could be extended to cursive scripts. Elarian et al. proposed a concatenative model for handwritten Arabic, though it relies on a database of pre-segmented characters and the concatenation procedure is specific to Arabic [5].

An alternative semi-supervised formulation of the problem assumes that there is a small labeled training set and a larger unlabeled training set. The main methodology involves propagating annotations from the labeled set to the unlabeled set through model prediction. Subsequent models then train as if the noisy predicted labels were ground truth annotations. Frinken et al. explored this method for isolated word image recognition in the framework of co-training, where a Hidden Markov Model (HMM) and a BLSTM model each made prediction that was used to further train the other model [6]. In a separate work, Frinken and Bunke use an ensemble of BLSTM networks for self-training, where high confidence ensemble predictions on the unlabeled data are subsequently used as ground truth to further train the ensemble. Ball and Srihari used a similar idea to adapt writer specific HWR models from a general model by iteratively updating segmented character prototypes after performing recognition on unlabeled data [2].

In this work, we propose a transfer learning methodology that allows us to train a HWR model for a *target* language for which we have no labeled images. Our method only requires a labeled training set of line images in a sufficiently similar *source* language, a trained Language Model (LM) in the target language, and a set of unlabeled images in the target language. A source language is sufficiently similar to the target language if the character sets of the two languages have a large overlap. For example, Latin based languages, such as English, French and Spanish, are all sufficiently similar because they all use the written Latin script. The LM can be obtained from digital text in the target language that is unrelated to the unlabeled images. Digital text for training a LM is much more commonly available than labeled handwriting images, so our methodology helps extend automated HWR to lower resource languages.

After training a HWR model on the source language, our proposed method begins a hybrid training procedure where training occurs on both source data and target data. There is no ground truth for the target data, so we combine the model prediction with the LM to produce a corrected prediction that we then use as ground truth.

We perform several experiments to analyze the behavior of our proposed transfer learning methodology for HWR. These experiments are performed using 4 datasets and 3 languages: English, Spanish, and French. We examine factors such as how long the source model is trained, LM decoding hyperparameters, and the proportion of source and target training used during hybrid training. In the best cases, we find that transferring produces

Character Error Rates (CER) nearly as low as those obtained by traditional supervised learning on the target data.

10.2 Language Transfer Learning

We formulate our problem as follows. Suppose we wish to obtain a trained HWR model for a target language Y that has no labeled training data available, but there are many unlabeled text line images in this language. We also have sufficient digital text in language Y, such that we can train a Language Model (LM). For another language X we have segmented text line images with corresponding ground truth transcriptions. Noting that languages X and Y have similar character sets, we want to use the data in both languages to produce the HWR model for language Y.

Though our discussion uses the term *language*, our methodology is also applicable to transfer learning HWR problems where there is a difference in domains (e.g. modern vs historical) or writers. We demonstrate this later by transferring between a modern English dataset and a historical English dataset.

10.2.1 Source Model Training

We begin transfer learning by training a state-of-the-art HWR model on the source language for which we have ground truth transcriptions. Fig. 10.1 shows our CNN-BLSTM architecture, which is similar to the model in [23], but introduces an auxiliary classifier and loss. This model learns high level features using convolution operations that are vertically collapsed to a 1D horizontal sequence of feature vectors that are fed to a 2-layer BLSTM. In the BLSTM, context is propagated both forwards and backwards along the sequence. Two separate frame-wise, linear character classifiers are each applied to the output of the CNN and the output of the combined CNN-BLSTM. Both classifiers are trained using Connectionist Temporal Classification (CTC) loss [8] which automatically aligns frame-wise outputs with the ground truth transcriptions.

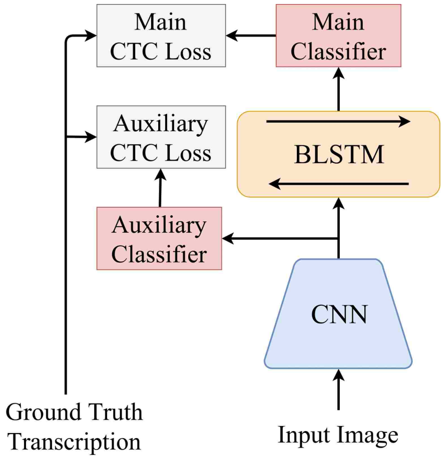


Figure 10.1: CNN-BLSTM architecture with auxiliary classifier and loss after the CNN.

The classifier that operates on the output of the CNN is considered an auxiliary classifier and it is discarded after the training procedure, meaning that the model outputs the predictions made by the main classifier that operates on the output of the BLSTM. We found that introducing an auxiliary classifier improves transferability of the model, likely because it forces the CNN visual features to be discriminative of characters themselves instead of depending on further processing from the BLSTM layers. When transferring between languages, the visual difference of some shared characters is small, so the CNN should be robust to the language difference. In contrast, the BLSTM considers the whole sequence, so it is more sensitive to transferring between datasets.

The precise architecture of our HWR model is based on the model presented in [23]. The size of the the input image is $W \times 60$, where W is the image width, which can dynamically vary. The CNN is composed of 6 convolutional layers with 3x3 learnable kernels, and there are 64, 128, 256, 256, 512, and 512 feature maps respectively for the 6 layers. We apply Batch Norm (BN) after layers 4 and 5 and 2x2 Max-Pooling (MP) with a stride of 2 after layers 1 and 2. After layers 4 and 6, we vertically collapse features by using 2x2 MP with a vertical stride of 2 and a horizontal stride of 1. To form the input for the BLSTM and for the CNN auxiliary classifier, we concatenate features in the same column to form a 1D horizontal sequence of 1024-dimensional feature vectors. The BLSTM has 2-layers each with 512 hidden nodes that have a 0.5 probability of node dropout. A linear classifier is applied to each time step to produce the final prediction, which is a probability distribution over characters at each timestep.

The model is trained using CTC loss over both the main classifier and the auxiliary classifier:

$$L(x, y) = \lambda L_{CTC}(\phi(x), y) + (1 - \lambda) L_{CTC}(\psi(x), y) \tag{10.1}$$

where x represents an input image, y, the corresponding ground truth transcription, L_{CTC} is the CTC loss [8], ϕ is the auxiliary CNN classifier, and ψ is the BLSTM classifier. We empirically set $\lambda = 0.25$ based on cross validation using validation data.

10.2.2 Language Model Decoding

The HWR model predicts each output character independently, and this may produce linguistically improbable sequences of characters. Decoding with a Language Model (LM) combines the individual predicted character probabilities with how likely sequences of characters occur together.

Our LM decoding implementation uses the Kaldi Speech Recognition Toolkit [16], which has been used previously in HWR models [3]. Similar to [22], we use a 10-gram character LM, which estimates $p(c_i | c_{i-1} c_{i-2} \dots c_{i-9})$ from digital text. Not all 10-gram character sequences are observed, so we smooth the empirical 10-gram distribution and employ backoff, where n-grams shorter than 10 are used to estimate the probability of infrequent 10-grams [4].

The decoding operation finds the most likely sequence of hidden states in a Hidden Markov Model (HMM), where the emission probabilities are determined by the HWR model and the transition probabilities are determined by the LM:

$$\hat{\mathbf{h}} = \arg\max_{\mathbf{h}} \prod_{i}^{N} p(h_i | h_{j<i}) p(x_i | h_i)^w \tag{10.2}$$

where $\mathbf{h}$ is the sequence of hidden states corresponding to characters, $h_{j<i}$ indicates all states prior to h_i, $\boldsymbol{x}$ is the observed data, and w determines the relative importance of the CNN-BLSTM and LM predictions. Because characters can span multiple output frames, we model each character using 3 states (corresponding to character start, middle, and end) as is commonly done in speech recognition [14]. The LM directly encodes the $p(h_i|h_{j<i})$ term, but the CNN-BLSTM outputs $p(h_i|x_i)$. Using Bayes Rule, we have

$$p(x_i|h_i) = \frac{p(h_i|x_i)p(x_i)}{p(h_i)} \tag{10.3}$$

We can estimate $p(h_i)$ by examining the CNN-BLSTM outputs, but $p(x_i)$ is unknown. Following Bluche et al., we approximate $\frac{p(x_i)}{p(h_i)} \approx p(h_i)^{-\alpha}$, where α is a hyperparameter [3]. An exact solution to Eq. 10.2 can be intractable, so in practice, we use a beam search which efficiently searches the state-space, but in some cases may not find the exact maximal sequence of characters.

10.2.3 Hybrid Training

Our hybrid training procedure leverages the recognition performance achieved by the source model on the source language to then learn recognition over the target language. The overall process is shown in Algorithm 1.

The main difference between hybrid and source training is in the data used for learning. During hybrid training, part of the data comes from the source dataset (typically 50%) with the rest coming from the target dataset for which there are no ground truth transcriptions. However, the training loss for hybrid training is the same as in source training (Eq. 10.1), which means that to train we need to provide some transcriptions for the target data.

We obtain target transcriptions by applying the LM of the target language to the predictions of the network. The intuition is that due to the similarity of the source and target languages, the predictions of the network will be much better than random, though

Algorithm 1 Hybrid Training Procedure

```
Input: Source Model $\mathbf{S}$, Source Data $\mathbf{X_S}$, Source Transcriptions $\mathbf{Y_S}$, Target Data $\mathbf{X_T}$, Target
    LM $\mathbf{Q}$
Output: Target Model $\mathbf{T}$
 1: Initialize $\mathbf{T}$ with weights of $\mathbf{S}$
 2: for $k = 0$ to $50$ do
 3:     // First update LM marginal probabilities to reflect $\mathbf{T}$
 4:     for $j = 0$ to $100$ do
 5:         Sample a minibatch $\mathbf{x_T}$ from $\mathbf{X_T}$
 6:         $\mathbf{P} \leftarrow \mathbf{T}(\mathbf{x_T})$
 7:         Use $\mathbf{P}$ to update $p(h_i)$ (used in Eq. 10.3)
 8:     end for
 9:     // Train $\mathbf{T}$ on source and target data
10:     for $j = 0$ to $100$ do
11:         Sample a minibatch $\mathbf{x_T}$ from $\mathbf{X_T}$
12:         $\mathbf{P} \leftarrow \mathbf{T}(\mathbf{x_T})$
13:         $\mathbf{y_T} \leftarrow \mathbf{Q}(\mathbf{P})$ (Eq. 10.2)
14:         Sample a minibatch $(\mathbf{x_S}, \mathbf{y_S})$ from $(\mathbf{X_S}, \mathbf{Y_S})$
15:         $\mathbf{x} \leftarrow \mathbf{x_T} \| \mathbf{x_S}, \ \mathbf{y} \leftarrow \mathbf{y_T} \| \mathbf{y_S}$
16:         Update $\mathbf{T}$ according to $L(\mathbf{x}, \mathbf{y})$ (Eq. 10.1)
17:     end for
18: end for
```

still quite poor at first. Applying the LM will improve the poor predictions to make better targets, which in turn helps the network to learn the target language better. We do, however, continue to train on source data to stabilize the learning process with real ground truth.

At the beginning of hybrid training, the model has never seen any instances of characters that are only part of the target language and will make incorrect predictions. However, the LM can correct some of these errors based on the context of surrounding correct predictions. For example, English words contain no accented characters, so a source model trained on English would never predict accented characters, but French and Spanish do use accents. The LM is able to correct the model predictions to include accents and thus introduce these characters into the ground truth so the model can learn to predict these characters in the future.

Because LM decoding depends on the marginal distribution of CNN-BLSTM outputs, $p(h_i)$ in Eq. 10.3, we need to periodically update this quantity. This is done in lines 3-7 in

Algorithm 1. In normal HWR model training, this is unnecessary because the LM is applied only as post-processing and not as part of the training process.

10.3 Datasets

In this work we use 4 datasets: IAM [13], Rimes [1], Rodrigo [18], and Bentham (2014 HTR competition) [17] collections. Each dataset is composed of a number of line images with corresponding ground truth transcriptions.

Rodrigo is a single author, 853 page Spanish manuscript written in 1545 with 20000 segmented line images. We used the first 750 pages as training data, the next 50 pages as validation data, and the remaining pages as test data. The annotations contain some meta information that we preprocessed to exclude. Some examples of this include symbols that indicate that whitespace should be inserted or deleted for correctness, i.e. the manuscript author did not conform to modern usage of whitespace.

The Bentham collection are the writings of the English philosopher Jeremy Bentham (1748-1832), though some images may be handwritten copies of his works produced by others [17]. For preprocessing, we deskewed the line images and performed height normalization. For IAM, we use the standard split, merging the two defined validation sets. For Rimes, there is only a defined train/test split, so we used a subset of the training data for a validation set.

Each image collection has different low level differences (e.g. color, texture), so we opted to binarize each dataset to eliminate those differences. This allows our analysis to focus on adapting to salient differences in language and style rather than on adapting to low level domain differences. For IAM, Rimes, and Rodrigo, we used Otsu binarization [15] but for Bentham, we used adaptive Wolf binarization [24] because it produced visually better binarizations.

To train the LMs for each dataset used in most experiments, we used the transcriptions of the training data. Though this corresponds to having an optimal LM for hybrid training, we also explore using LMs trained on unrelated data. For these LMs, we sampled 50000 sentences from the United Nations proceedings subset of the Europarl machine translation dataset [11] in Spanish, English, and French.

To obtain the character classes predicted by our models, we take the union of the character sets of each dataset. Because of this, during source model training, the classifiers output distributions over all characters, not just those characters contained in the source dataset. This way if the target dataset has additional characters, we do not need to modify the classifiers before hybrid training.

10.4 Experimental Results

In the following experiments, we use the following protocol. For source models, we trained 4 models for each dataset for 10 epochs using the ADAM optimizer to perform weight updates [10]. We then selected the best model using the Character Error Rate (CER) on the validation set after performing LM decoding using the dataset-specific LM. All reported numbers for source models are on the designated test splits for each dataset. These source models were used as the initial models in all hybrid training experiments, except where noted.

For hybrid training, we also trained 4 models where each hybrid model is initialized with the weights learned on the source dataset. Hybrid models are trained for approximately 12000 weight updates using mini-batches of 8 images, where mini-batches contain both source and target images. To report metrics, we select the best model based on the validation set for the target data and then evaluate this model on the target data. While in practice this is not feasible because target data would not have ground truth transcriptions, this allowed us to fairly compare different methods of hybrid training. We leave a method for selecting the best model without using ground truth as future work.

10.4.1 Source Model Evaluation

Table 10.1 shows the CER of source models when evaluated on each dataset. As expected, source models obtain low CER when the test data matches the training data and high CER

Table 10.1: CER of source models evaluated on each dataset.

		Test Data			
		Bentham	IAM	Rimes	Rodrigo
Train Data	Bentham	4.7	45.5	43.3	26.1
	IAM	27.8	8.4	16.1	24.3
	Rimes	35.0	24.6	4.9	34.4
	Rodrigo	69.5	67.0	67.1	6.8

when there is a mismatch. Though this result may be obvious, it demonstrates the need for our hybrid training methodology in order to transfer models from one language to another. We also note that even though IAM and Bentham are both English datasets, models trained on one do not perform well on the other and have need of transfer learning.

The CERs obtained are competitive when compared with previous results reported in the literature. For example, [3] reports CER of 3.9 and 3.8 for IAM and Rimes respectively, while we achieve 8.4 and 4.9 CERs. In [7], a CER of 3.0 is reported on the Rodrigo dataset, though this number is not directly comparable to our reported results because they use a different data split and transcription preprocessing. Additionally, we binarized our data for transferability and generally CNN-BLSTM models perform better when using grayscale inputs. The best CER on Bentham reported in the 2014 ICFHR HTR compeition is 5.0 for the restricted track [17]. Also, our reported numbers are on source models that have not trained to convergence (this improves hybrid training) but further training of the source models produces 1-2% lower CERs.

10.4.2 Hybrid Training

In hybrid training, we varied 4 factors to gain a better understanding of the sensitivities of the method:

- Length of source model training time
- Proportion of source and target data

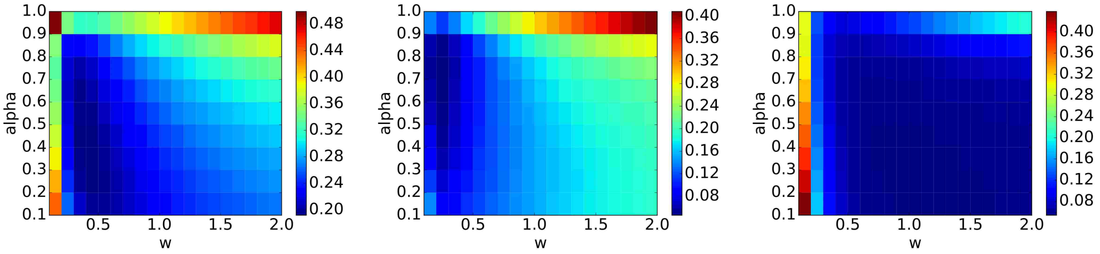
Table 10.2: CER of hybrid trained models for all language pairs across a variety of experimental settings.

Experimental Conditions				Source: Bentham			Source: IAM			Source: Rimes			Source Rodrigo			
Source	LM	LM	Amount	Transfer to			Transfer to			Transfer to			Transfer to			Avg.
Epochs	data	w / α	Source	IAM	Rim.	Rod.	Ben.	Rim.	Rod.	Ben.	IAM	Rod.	Ben.	IAM	Rim.	(no Rod.)
10	Train	0.4 / 0.5	50%	**9.2**	**7.3**	**12.0**	8.2	6.3	13.0	8.8	8.2	**12.6**	64.5	77.2	35.1	8.0
50	Train	0.4 / 0.5	50%	9.8	7.5	13.3	8.6	6.3	13.5	9.3	8.4	13.4	70.3	74.3	70.1	8.3
10	Train	0.4 / 0.5	75%	9.3	7.6	13.2	**7.5**	**5.7**	12.1	9.4	**7.8**	12.8	**59.4**	**64.7**	33.4	**7.8**
10	Train	0.4 / 0.5	25%	10.7	7.8	13.1	9.3	6.3	**11.9**	**8.5**	8.7	**12.6**	71.8	73.8	37.5	8.6
10	Train	0.8 / 0.4	50%	12.1	8.1	100.0	8.2	6.7	97.0	8.9	8.9	13.5	67.2	80.1	18.3	8.8
10	Train	1.2 / 0.3	50%	30.8	9.3	91.7	11.1	7.2	79.1	10.3	12.4	96.0	66.7	74.1	**11.1**	13.5
10	None	-	50%	54.3	69.4	38.4	27.3	17.4	27.0	58.7	22.1	44.4	100.0	84.4	98.4	41.5
10	Europarl	0.4 / 0.5	50%	32.6	80.5	85.8	13.4	12.0	23.6	20.7	18.6	36.8	99.8	99.1	99.8	29.6
Source Model - no Hybrid Training				45.5	43.3	26.1	27.8	16.1	24.3	35.0	24.6	34.4	59.5	67.0	67.1	32.1

- Data used to train the LM
- The w and α LM parameters

Table 10.2 shows the CER after hybrid training for all language pairs for all experimental settings. Here we explain the column semantics of Table 10.2. Source Epochs indicates how long source models were trained before hybrid training began. We also varied the data used for LM training - either the ground truth training set transcriptions, Europarl corpus subset, or no LM was used. The next 2 columns respectively indicate the LM hyperparameters and percentage of source data used in hybrid training. Remaining columns indicate Source-Target dataset pairs, where the first header row indicates the source language with target languages listed below. For example, the first data column is Bentham as the source with IAM as the target. The last column shows the average performance of the 6 language pairs involving Bentham, IAM, and Rimes. For this average, we excluded Rodrigo because of the extremely high CERs of unsuccessful transfers, which would dominate the average. For comparison, the last row shows performance of the source models before hybrid training, i.e. the off-diagonal entries of Table 10.1.

Considering the first 4 rows of Table 10.2, pairwise transfers between Bentham, IAM, and Rimes are extremely successful, achieving CERs near to those obtained with full supervised training in some cases. It is interesting that while these three datasets can transfer to Rodrigo with CERs of about 13%, the reverse is not true. Only Rodrigo to Rimes hybrid training managed to significantly improve the CER over the source model, achieving 11.1%



(a) Bentham (b) Rimes (c) IAM

Figure 10.2: LM parameter search visualized as a heatmap of resulting CER. We used the IAM source model and evaluated on Bentham, Rimes, and IAM validation sets for many values of α, w. For Bentham and Rimes, the optimal $\alpha \approx 0.5$, $w \approx 0.4$. For IAM, the optimal $\alpha = 0.3$, $w = 1.2$.

CER under one set of experimental conditions, though this language pair appears sensitive to variations in experimental conditions.

When the source model is trained to convergence, i.e. trained for 50 epochs instead of 10, CER on the source data improves by about 1-2% (data not shown), but the CER after transferring increases for all language pairs except one. The average CER increases by 0.3%. This is because after training for so long, the models can overfit the source data and may have difficulty unlearning factors unique to the source dataset.

Next we examined what proportion of source and target data is used during hybrid training. Overall, using 75% source data produces an average CER of 7.8% vs 8.0% for equal proportions and 8.6% for 25% source data. We also note that the optimal percentage of source data varies by language pair. Because the target labels provided by the LM are not always correct, the model can diverge if it is presented with too many poor quality target labels. Source data helps stabilize hybrid training, so using a larger proportion of source data may make training more stable.

Next we examined the LM parameters w and α used for LM decoding (Eqs. 10.2,10.3). We determined our default values of $w = 0.4$ and $\alpha = 0.5$ by cross validation to optimize the CER of source models evaluated on the datasets that they were not trained on (i.e., the off-diagonal entries of Table 10.1). For example, Fig. 10.2 shows heatmaps for the source IAM model evaluated on Bentham and Rimes. When evaluating the IAM model on Bentham or Rimes, we see better performance when $w \approx 0.4$ and $\alpha \approx 0.5$, but when we evaluate on IAM, $w = 1.2$ and $\alpha = 0.3$ perform best. We saw a similar trend when evaluating the Bentham source model on the other datasets.

A similar trend also holds for hybrid training. Our default parameters of $w = 0.4$, $\alpha = 0.5$ achieved an average CER of 8.0%, which is lower than 8.6% with $w = 0.8$, $\alpha = 0.4$ and 13.5 with $w = 1.2$, $\alpha = 0.3$. Also, transferring to Rodrigo becomes unsuccessful when using these alternate parameters. However, it is interesting that these parameter settings greatly improve transfer from Rodrigo to Rimes (achieving 18.3 and 11.1 CER). Thus the optimal LM hyperparameters vary based on the language pair, and unfortunately, they cannot be estimated by cross validation in a real setting as cross validation relies on the ground truth for the target language.

We conclude our experiments by varying the data used to train the LM. If we do not apply LM decoding during hybrid training (or equivalently use a LM where all sequences of characters are equally likely), we see that some language pairs improve over the source model performance, though some do not improve or get worse. When we use the Europarl trained LMs, we see degraded performance with respect to the LMs trained on the dataset training sets, but this is expected to some degree. The Europarl corpus uses very formal language, and the modern Spanish is very different from the historical Spanish used in the 1545 Rodrigo manuscript. Transferring Bentham to IAM and vice versa, using the Europarl English LM greatly improves CER compared to using no LM at all. The same is also true for IAM and Rimes.

10.5 Conclusion

In this work we proposed a methodology that trains HWR on a target language without using any labeled data in that language. It does so by leveraging labeled images in a closely related source language and a language model in the target language. After training a source model, we train on both the source and target data, inputting target labels using the current model predictions decoded by the LM. We demonstrate that our approach is successful on many pairs of languages using the IAM, Rimes, Bentham, and Rodrigo datasets. We explored the design choices of our hybrid training approach and make conclusions about the LM training data, LM hyperparameters, amount of source data in hybrid training, and length of source model training.

References

[1] Emmanuel Augustin, Jean-marie Brodin, Matthieu Carré, Edouard Geoffrois, Emmanuèle Grosicki, and Françoise Prêteux. RIMES evaluation campaign for handwritten mail processing. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2006.

[2] Gregory R Ball and Sargur N Srihari. Semi-supervised learning for handwriting recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 26–30. IEEE, 2009.

[3] Théodore Bluche, Hermann Ney, and Christopher Kermorvant. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In *International Conference on Statistical Language and Speech Processing*, pages 199–210. Springer, 2014.

[4] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, Elsevier, 1999.

[5] Yousef Elarian, Irfan Ahmad, Sameh Awaida, Wasfi G. Al-Khatib, and Abdelmalek Zidouri. An Arabic handwriting synthesis system. *Pattern Recognition*, 48(3):849–861, Elsevier, 2015.

[6] Volkmar Frinken, Andreas Fischer, Horst Bunke, and Alicia Foornes. Co-training for handwritten word recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 314–318. IEEE, 2011.

[7] Emilio Granell, Edgard Chammas, Laurence Likforman-Sulem, Carlos D. Martínez-Hinarejos, Chafic Mokbel, and Bogdan-Ionuţ Cîrstea. Transcription of Spanish historical handwritten documents with deep neural networks. *Journal of Imaging*, 4(1):15, Multi-disciplinary Digital Publishing Institute, 2018.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 369–376. JMLR, 2006.

[9] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, IEEE, 2009.

[10] Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *International Conference on Learning Representation (ICLR)*, ArXiv, 2014.

[11] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit*, volume 5, pages 79–86. International Association for Machine Translation, 2005.

[12] Do-Hoon Lee and Hwan-Gue Cho. A new synthesizing method for handwriting korean scripts. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(01): 45–61, World Scientific, 1998.

[13] U. V. Marti and Horst Bunke. The IAM-database: An English sentence database for offline handwriting recognition. *IJDAR*, 5(1):39–46, Springer, 2002.

[14] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[15] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, IEEE, 1979.

[16] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *Workshop on Automatic Speech Recognition and Understanding*. IEEE, December 2011.

[17] Joan Andreu Sánchez, Verónica Romero, Alejandro H. Toselli, and Enrique Vidal. ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 785–790. IEEE, 2014.

[18] Nicolás Serrano, Francisco Castro, and Alfons Juan. The RODRIGO database. In *Conference on International Language Resources and Evaluation (LREC)*, pages 19–21. European Languages Resources Association (ELRA), 2010.

[19] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962. IEEE, 2003.

[20] Cheng-Huang Tung, Yuan-Jong Chen, and Hsi-Jian Lee. Performance analysis of an OCR system via an artificial handwritten Chinese character generator. *Pattern Recognition*, 27(2):221–232, Elsevier, 1994.

[21] Tamás Varga and Horst Bunke. Perturbation models for generating synthetic training data in handwriting recognition. In *Machine Learning in Document Analysis and Recognition*, pages 333–360. Springer, 2008.

[22] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *Interna-*

tional Conference on Frontiers in Handwriting Recognition (ICFHR), pages 228–233. IEEE, 2016.

[23] Curtis Wigington, Seth Stewart, Brian Davis, Bill Barrett, Brian Price, and Scott Cohen. Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 639–645. IEEE, 2017.

[24] C. Wolf, J.-M. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. In *International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1037–1040, 2002.

Chapter 11

Conclusion, Contributions, and Future Directions

This dissertation has presented several proposed deep learning models to solve a variety of problems in the field of document image analysis. The document analysis community has shown that in many instances, taking an off-the-shelf deep model and finetuning it on a document related task often outperforms handcrafted approaches. However, such models are not designed for the domain of document analysis due to the unique properties of document images. The models proposed in this dissertation have been designed for their respective tasks and outperform their off-the-shelf counterparts.

We briefly review the contributions of the dissertation. Chapter 2 presents an up-to-date review of historical document image binarization. A novel FCN architecture and loss function for binarization is proposed in Chapter 3. Generating realistic synthetic data for the purpose of training deep networks for binarization is the subject of Chapter 4. Chapters 5 and 6 both propose models for segmenting documents from their surrounding background, and Chapter 7 deals with table structure decomposition. Chapter 8 is a detailed empirical study of deep networks for document image classification. Chapter 9 presents a state-of-the-art model for font recognition. Transfering a handwriting recognition model between languages is the subject of Chapter 10.

This dissertation concludes with a discussion of future research directions.

11.1 Directions for Future Research

While this dissertation explores a number of different problems in document image analysis, there are several overall directions for future research.

One approach would be to keep refining deep learning models to be better suited for each document related task. This is one of the focal points of this dissertation and the body of this work has shown that such an approach leads to improved performance. For example, including the textual content of a document for classifying page images with CNNs (Chapter 8) is likely to improve performance, but the best way to fuse the language and image modality is the subject of active research. In binarization, combining deep models with some of the other proven techniques is a promising direction to pursue. For example, none of the deep models reviewed in Chapter 2 perform image preprocessing. These deep models are trained to directly classify pixels, but instead training them to predict local thresholds could lead to smoother output images, explainable decisions, and improved performance.

There are also a large number of document understanding tasks that this dissertation has not covered. Such tasks include, among others,

- Bleed-through detection and removal
- Document dewarping
- Coarse segmentation of the document into paragraphs, tables, figures, etc
- Fine segmentation into text lines and words
- Vector graphics and figure recognition
- OCR
- Language identification
- Writer identification
- Word spotting and document indexing/retrieval

While some deep models have been proposed for many of these tasks, continuing to adapt the learning models to such tasks and their domains of interest is a subject of on-going research.

While proposing new models to solve individual tasks has lead to large improvements, in practice, few of these tasks are actually solved in isolation. In real word systems, these tasks need to be performed in sequence (e.g. binarization, segmentation, then recognition) and errors made in previous steps affect the performance of all the down stream processing steps. What deep models enable is end-to-end learning, where the errors made by later models can be fed back to improve earlier models. One example of this is the Start-Follow-Read model, where one model localizes the start of text lines, another model performs text line dewarping, and the last model performs recognition on the dewarped handwriting [1]. The recognition errors from the last model are used to improve both the start of line localization as well as the dewarping performed.

There are many other such sets of related tasks that can help inform each other's performance. Identifying the language, script, and font of text can aid in character and word recognition, but the errors made in recognition can be informative for training language or font identification models. Both text line segmentation and page localization (Chapters 5 and 6) can inform text binarization by removing background noise far from text. The majority of false positive errors made on the DIBCO 2018 dataset are due to the surroundings of the document rather than the document noise. End-to-end learning of such tasks could improve performance on each component task, but has been relatively unexplored in the literature.

Multi-task learning is another way to pool information across tasks without chaining their inputs and ouputs together. For example, a model that simultaneously predicts text binarization and scribal script classification might learn to make script classification decisions based more on character shapes learned from binarization rather than the correlated noise factors analyzed in Chapter 9. Joint learning of writer identification and handwriting recognition might lead to handwriting recognition that learns to correctly transcribe a wider variety of handwriting styles. The examples of pairs of tasks suitable for multi-task learning are numerous.

One of the key issues of improving deep learning systems in document analysis is the lack of large sets of richly labeled data. While end-to-end learning and multi-task learning allow for data from other tasks to be used and are a partial solution to this problem, the reality is that most document analysis datasets are small. This is in part because the need for large training sets for deep models is relatively new. One broad avenue for creating larger datasets is to create large synthetic datasets using unsupervised learning. Chapter 4 explored this direction for binarization for historical documents, where the difference between synthetic and real data mainly is in the local noise distributions. Such low level modeling is relatively easy to perform using current deep convolutional models, but generating realistic synthetic data for higher level tasks, such as realistic layouts for segmentation or character recognition, may prove more challenging.

References

[1] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. Start, follow, read: End-to-end full-page handwriting recognition. In *European Conference on Computer Vision (ECCV)*, pages 367–383. Springer, 2018.