INTERACTIVE MACHINE LEARNING FOR WORD RECOGNITION ON DAMAGED HANDWRITTEN DOCUMENTS

By
Jack Bandy

Director of Project: Brent Seales

Director of Graduate Studies: Miroslaw Truszczynski

Date: April 4, 2018

MASTER'S PROJECT

Jack Bandy

The Graduate School
University of Kentucky
2018

INTERACTIVE MACHINE LEARNING FOR WORD RECOGNITION ON DAMAGED HANDWRITTEN DOCUMENTS

MASTER'S PROJECT

A document submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Arts and Sciences at
the University of Kentucky

By
Jack Bandy
Lexington, Kentucky

Director: Dr. Brent Seales, Professor of Computer Science
Lexington, Kentucky 2018

ACKNOWLEDGMENTS

Acknowledge people/things here

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Related Work

For several decades, researchers have been developing methods for automated character and word recognition. These methods take some photograph(s) of printed or handwritten text as input, and produce a transcript of that text as output. This section provides a brief summary of methods which have influenced the course of this research area, including advances in handwriting recognition, printed text recognition, and handwritten word spotting.

The nomenclature for these related tasks can be somewhat inconsistent in the literature. For the purposes of this paper, "handwriting recognition" differs from "handwritten word spotting" in that the former aims to create full transcriptions while the latter merely locates and/or recognizes instances of a given word within a document. "Printed text recognition," although it uses many of the same methods, refers to projects that examine machine-printed texts. The consistency of character representations and thus word representations drastically changes the task, so a distinction is necessary.

Text Recognition

"Text recognition" here refers to recognizing *printed* texts, not handwritten texts. From a technical standpoint, automatic text recognition is the task of turning an image into the text within the image.

Object character recognition (OCR) on scans of printed documents has seen success since as early as the 1980s [2, 3]. Due to the consistency of letter shapes and sizes, fairly simple techniques for pattern recognition could accurately classify characters in the same font family. However, as early as 1987, font and size constraints were no longer needed. The authors of [4] demonstrated a system that accurately classified mixtures of dissimilar fonts of varied sizes.

Gradually, more and more constraints were eliminated. After [4] removed the need for font and size assumptions, the race was on to eliminate constraints such as alignment, color, contrast, and more. Eventually, the task of printed text recognition was one that could be done "in the wild," [5, 6, 7] with essentially no assumptions about the nature of the text. Especially important for "in the wild" recognition was eliminating the segmentation step, as in [8], such that regions of text could be found without a processing phase devoted to localization. The ideal system, then, would be able to recognize text in any image in which a human could see text.

An important benchmark dataset for this kind of text recognition is Street View Text (SVT) [9]. SVT was harvested using pictures from Google Street View, and thus contains a heterogeneous collection of word images with a variety of fonts, colors, backgrounds, and more. (Despite the variations, word images in this set do not include handwritten characters.) The SVT dataset was released in 2010, and by

2012, [6] demonstrated state-of-the-art performance for both character recognition and word recognition by training on images from the dataset. The high degree of accuracy was achieved via unsupervised feature learning and convolutional neural networks.

Even before 2012, many researchers realized that convolutions provide an ideal mechanism for recognizing the shapes of different letters. Others have taken more general approaches to text recognition via CNNs [6, 7], . The network architectures from these papers are, on the whole, restrictively large, whereas both architectures from my experiments were able to run on my laptop.

Handwriting Recognition

Although modern methods for printed text recognition overlap methods for handwriting recognition, especially with CNNs for "in-the-wild" recognition, the convergence happened after years of parallel research. Handwriting recognition can be divided into two major categories, "online" handwriting recognition and "offline" handwriting recognition. In the former, software tracks the location of a writing utensil as a user moves it across some surface to produce letters and words, and the precise location and motion of the utensil helps reveal the intended writing. For example, UNIPEN [10], a benchmark dataset for online handwriting recognition, includes "pen trajectory" data that specifies when and where the pen touched down and lifted up, as well as the coordinates for the path of the pen.

More relevant to this project is the task of offline handwriting recognition, in which the input comprises only a picture of the handwriting and no additional information about its creation. A canonical example of the text recognition task is the MNIST dataset [11]. MNIST comprises grayscale images of individual handwritten digits, 0 to 9, and the objective is to classify each image into the digit written inside of it. Machine learning researchers have been using this task as a benchmark for several decades [12], with error rates well below 1% since 2003 [13].

Projects using MNIST and similar datasets are premised upon many constraints. For example, a very small vocabulary or character set could be recognized if they were properly aligned and segmented, but as soon as a text ventured outside those constraints (variations on letters, misspelled words, new characters, etc.), the system would falter. Even moderately successful recognition on unconstrained datasets did not exist until the early 2000s.

This changed with the use of hidden Markov Models (HMMs) [14, 15, 16]. With statistical models built for specific languages, character and word recognition accuracies improved to over 85% (varying with respect to the test corpus). More impressively, these results came on *unconstrained* texts.

While HMMs made the way for unconstrained datasets, many demonstrations were still using the IAM dataset [17], an ad-hoc database for researchers. In other words, *truly* unrestricted handwriting recognition was still a long way off even after the strides made by HMMs. Moving forward, a collection of George Washington letters became the de-facto standard. This dataset comprised hundreds of manuscript pages from the Library of Congress, handwritten by George Washington's secretaries.

In the mid-2000s, even state-of-the-art HMM methods yielded word error rates around 50% on datasets such as the George Washington collection. But around this time, researchers began taking a new angle at the problem. Specifically, projects focused on the process of "handwriting retrieval," rather than attempting complete transcriptions. Such projects allow users to query a dataset of images for a given word, and essentially scans the images for visual matches of that word. For example, [18] presents a word retrieval system that achieves 63% mean average precision scores on the George Washington collection.

In [19], this approach is formalized as a viable way to generate a searchable index of handwritten papers. Their method of "wordspotting" turns the search problem into a clustering problem, where word images that are "closest" to the query word are considered matches. Wordspotting is considered more thoroughly in the following section, however it is crucial to note that this approach eliminated the need for recognizing words before retrieval. In other words, matching is done in real-time.

Building upon the success of wordspotting techniques and HMMs, [20] takes a step further and first detects *characters* in a word, before inferring a word using an ensemble of HMMs. This approach allowed the recognition of words that were never seen during training, and established new standards for the George Washington dataset.

By the time ensemble HMMs came onto the scene, neural networks were already penetrating the field of handwriting recognition [21]. By 2010, advanced techniques such as bidirectional long short-term memory (BLSTM) were successfully applied to wordspotting [9] and outperformed other methods. Finally, recurrent neural networks [22] eliminated the need for word segmentation in addition to improving state-of-the-art performance on recognition tasks.

More recently, convolutional neural networks (CNNs) have become the state-of-the-art approach for text recognition on handwritten documents [23, 24]. Many of these approaches overlap text recognition methods mentioned in the previous section, and in fact, recent neural networks are designed to recognize both printed text and handwritten text.

Word Spotting on Damaged Handwritten Documents

In this section, the scope of related projects is narrowed down from all handwriting recognition systems, and I examine research related to word spotting on historical documents.

As previously mentioned, [19] formalized the idea of wordspotting. However, the concept was originally proposed in [25], which clustered similar words to be annotated by users, and reported success for documents written by a single person in high-quality handwriting.

1.2 Motivation

On the surface, optical character recognition, word recognition, and handwriting recognition appear to be solved problems. As detailed in the previous section, the

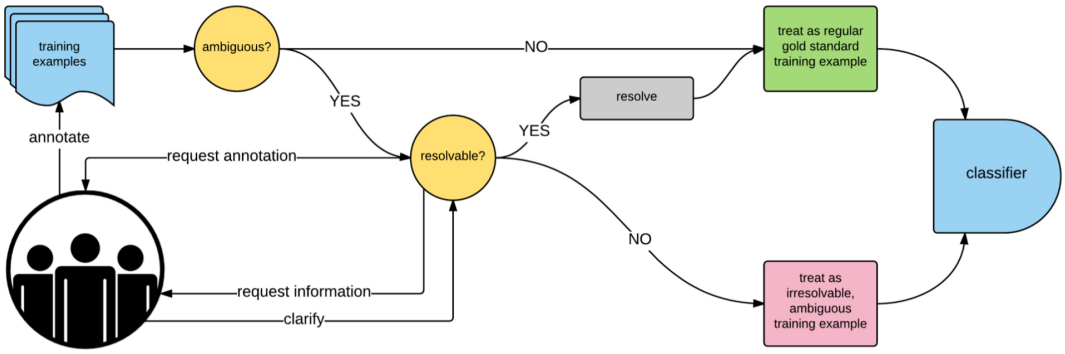


Figure 1.1: A schematic diagram from [1] for semi-automated transcription. This framework is designed especially for resolving ambiguity, which is not the focus of my project. Nonetheless, this diagram depicts a collaborative framework betwee machine learning and human annotation.

explosion of machine learning research in recent years has led to drastic improvements in performance on these tasks, and many advancements have even found their way to consumer products. For example, everyday software allows users to search within scans or photographs of printed typeface, and note-taking software can now interpret penmanship that would be indecipherable to many human readers.

However, the process of transcribing ancient documents presents a niche area of text recognition which is not addressed well by standard approaches. Many historical documents, including those reviewed in this project, were meticulously transcribed with legibility comparable to typeface, suggesting that automated transcription would be straightforward. But over time, these documents have incurred damage of all different kinds. The characters originally may have looked like typeface, but after hundreds of years of human handling, physical corrosion, chemical decay, and other processes, reading certain parts of these documents is an arduous task even for skilled textual analysts. For such cases, neither fully human transcription nor fully automated transcription is ideal.

While fully manual transcription is the most accurate solution, it is incredibly time-consuming for larger documents. Moreover, on damaged documents, skilled papyrologists are required to decipher texts. In short, human transcription is often prohibitively costly in terms of time and skilled personnel.

A fully automated transcription algorithm may successfully transcribe certain portions of a historical document, but the damaged portions can distort the algorithm's output to the point of being unusable. This is especially true in cases where letters are literally missing. This is especially true for OCR algorithms which assume constant width, spacing, and more within a document.

An ideal solution would leverage automated transcription for the undamaged portions, and allow a human reader to fill in any gaps. A framework for this hybrid approach is presented in [1], which also provides an elegant schematic of the approach in Figure 1.1.

I refer to this as semi-automated transcription. This project presents a pipeline for semi-automated transcription, blending the irreplicable abilities of the human eye with the efficiency and scalability of character recognition algorithms.

1.3 Contributions

The methods used in this project borrow heavily from methods in the aforementioned research areas, including keyword and character spotting [26, 22], word recognition [20], and handwriting recognition [27, 28].

Nonetheless, this project makes two notable contributions to the area of handwritten text recognition, both concerned with the challenges of damaged words. First, a framework for semi-automated transcription is detailed and implemented, in which damaged words could be labeled by an expert and seamlessly integrated into an otherwise automated word spotting process. The second contribution is an approach for virtually restoring damaged or low-quality words into a representation that could be recognized automatically.

An Interactive Approach to Word Spotting

A semi-automated approach to word spotting utilizes user-provided labels of words in a given document. Essentially, a group of users generate a training set, which allows the system to recognize most occurrences of a given word in the training set. If a distorted occurrence of the word exists, the system will leave it unlabeled and leave the word image in the pool of images to be labeled by users.

A Technique for Virtual Ink Restoration

Several techniques .

Chapter 2 Methodology

2.1 Preprocessing

For the George Washington dataset, segmented and binarized word images are already provided. For the Wycliffe dataset, several preprocessing steps must be taken to get from raw images of the manuscript to segmented, binarized word images suitable for word spotting.

Alignment

The first step, visualized in figure 2.1, involves rotating the original photographs so that text columns are vertically aligned. Rotation varies depending on where the page existed in the binding and which side of the book it was on.

Columns were cropped identically on each page, based on the assumption that more precise segmentation would take place in the line segmentation and word segmentation algorithms. The key in column cropping is to create images that only contain text from one column. The amount of margin outside the column need not be precise, however, to allow for key assumptions in the binarization phase, it must not contain anything besides paper.

Binarization

Once the column images are cropped, the RGB image is flattened into a single grayscale channel. The image is then inverted so that the text is white and the background dark gray. To remove the gray background, the system uses a thresholding algorithm implemented by OpenCV [**?**]. Because lighting and coloring varies across manuscript pages, a global threshold would lead to noisy and inconsistent background removal. Instead, a threshold should be calculated individually for each page.

Because the column image is assumed to only contain ink and paper, a histogram of values in the column image should be bimodal (the two peaks representing the approximate value of an ink pixel and a paper pixel). Otsu's binarization algorithm [**?**] takes advantage of this bimodal distribution. It works by choosing a threshold value in between the two peaks that minimizes the variance within the two "classes," an ideal method for these manuscript pages.

Segmentation

After column images are binarized, the next step is to split the column into its individual lines and words.

The Wycliffe New Testament is aligned and spaced with remarkable consistency. So, the segmentation involved plotting a vertical projection profile of a page image to determine the location of individual lines of text. To segment the line of text into

Figure 2.1: A sample of the original photographs of the Wycliffe New Testament Manuscript. In the preprocessing phase, these images must be aligned and cropped into separate columns.

individual words, in this case, a simple threshold on the horizontal projection profile provided sufficient accuracy. Because of the alignment, the intensity vector across the line had consistent valleys at the spaces in between each word.

Damage Simulation

The George Washington dataset, which is used for baseline testing, exists in fairly high quality.

Feature Extraction

Two methods were used for feature extraction. Histogram of oriented gradients (HOG) features were used as a baseline, and extracted using a scikit-image implementation [**?**]. The second method, the VAE's encoded representation of images, is detailed in the following section.

2.2 Variational Autoencoder

The purpose of the VAE is to learn an encoded representation of the word images without the need for ground truth label. The general structure of a VAE trains by encoding its input then attempting to recreate it using the decoder. Backwards propagation of error occurs based on how similar the decoded image is to the original input image. Thus, a successful encoding allows the decoder to accurately recreate the input image. This process is visualized in figure **??**.

Figure blank shows. The experiment used two different architectures to determine the effect of expanding or contracting the size of the encoded layer, both were implemented using Keras [29] with TensorFlow [30] used as the backend.

2.3 Clustering

To determine which word images contain the same word, clustering is performed on the encoded representation of the word images. The experiment tried two different clustering schemes: k-means and agglomerative. K-means is a popular centroid-based approach which iteratively refines the approximated center of each cluster [31, 32]. On the other hand, agglomerative clustering works by repeatedly merging the two closest clusters, resulting in a dendrogram with a single "cluster" at one end and n "clusters" at the other, where n is the number of data points.

Agglomerative clustering provides a natural fit for clustering encoded word images. Intuitively, each step groups the two most similar images into the same cluster. So instead of defining the number of clusters before the algorithm starts, the algorithm can stop when the two most similar images differ by a given amount.

2.4 Priority Queue

In the current approach, images are labeled in the order which they appear. However, this can be modified depending on the need of the given task. For example, if a general understanding of the document's contents is more important than a word-for-word transcription, the system could request labels for frequently occurring words. Or, for partially damaged documents, the system could request labels for "misfit" words that are particularly confusing to the encoder or to the clustering algorithm.

2.5 Providing Labels

Labels are provided in a simple graphical user interface which displays the original word image (before inverting it and removing the background). The interface, built using TKInter, allows a user to type in a label for the word image, skip to the next word, or flag the word image as needing re-segmentation.

In the experiments, a simulated oracle was used for labeling. Because the word images were sorted in order of appearance in the text, the oracle simply labeled the image using the word at the corresponding index of the transcript.

2.6 Transcription

A transcription of the full set of word images can be given at any time. Each word image must be encoded by the network. The encoded representation is used to determine which cluster of images the word belongs to. Once a word image is clustered, transcribing it is simply a matter of applying the label for that cluster. If the cluster is unlabeled, words in that cluster will show up in the transcript as "unknown."

Chapter 3 Evaluation

There were two goals in the evaluation phase. First, to determine whether the VAE's encoded representation of images could be used as features in the clustering phase of word spotting. The second goal was to determine whether the VAE could reconstruct damaged words so as to accurately classify them during word spotting.

3.1 Data Background

Two datasets were used to evaluate the interactive approach to word recognition and ink restoration.

George Washington Dataset

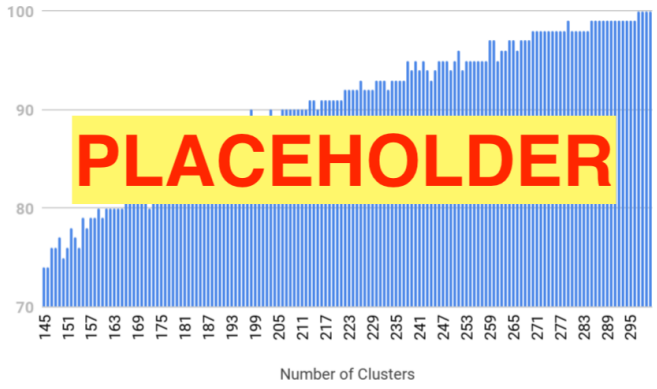
The George Washington Dataset is enormously popular for evaluating handwriting recognition.

The subset of data used in this paper is from work done by [**?**]. The authors provide word segmentation, ground truth, and normalized images for 20 pages of the George Washington letters.

Wycliffe Dataset

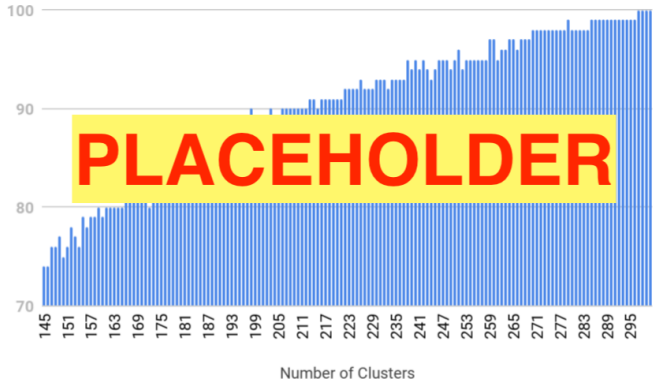
3.2 Classification on George Washington Dataset

Results from experiments on original word images.



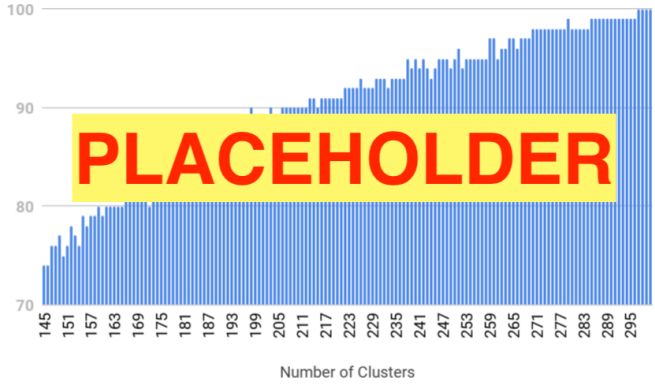
3.3 Classification on "Damaged" George Washington Dataset

Results from experiments on damaged words.



3.4 Wycliffe New Testament

Results from experiments on the Wycliffe dataset. First four pages or so?



Chapter 4 Conclusion

4.1 Challenges

What I ran into.

4.2 Findings

What I found.

4.3 Future Work

Suggestions for future work.

Chapter 5 The Example Chapter

5.1 The Example Section

Math goes here.

Here's a figure

Figure 5.1: A Simple Figure

Here	is
a	table

Table 5.1: A Simple Table

Bibliography

[1] Mike Schaekermann, Edith Law, Alex C Williams, and William Callaghan. Resolvable vs. irresolvable ambiguity: A new hybrid framework for dealing with uncertain ground truth. In *1st Workshop on Human-Centered Machine Learning at SIGCHI*, 2016.

[2] J Mantas. An overview of character recognition methodologies. *Pattern recognition*, 19(6):425–430, 1986.

[3] VK Govindan and AP Shivaprasad. Character recognition?a review. *Pattern recognition*, 23(7):671–683, 1990.

[4] Simon Kahan, Theo Pavlidis, and Henry S Baird. On the recognition of printed characters of any font and size. *IEEE Transactions on pattern analysis and machine intelligence*, (2):274–288, 1987.

[5] Ray Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[6] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[7] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[8] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545–555, 2015.

[9] Kai Wang and Serge Belongie. Word spotting in the wild. In *European Conference on Computer Vision*, pages 591–604. Springer, 2010.

[10] Isabelle Guyon, Lambert Schomaker, Réjean Plamondon, Mark Liberman, and Stan Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 29–33. IEEE, 1994.

[11] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[12] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Lawrence D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 77–82. IEEE, 1994.

[13] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12):971–981, 2004.

[14] U-V Marti and Horst Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. In *Hidden Markov models: applications in computer vision*, pages 65–90. World Scientific, 2001.

[15] Horst Bunke, Samy Bengio, and Alessandro Vinciarelli. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE transactions on Pattern analysis and Machine intelligence*, 26(6):709–720, 2004.

[16] A El-Yacoubi, Michel Gilloux, Robert Sabourin, and Ching Y. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[17] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[18] Toni M Rath, R Manmatha, and Victor Lavrenko. A search engine for historical manuscript images. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 369–376. ACM, 2004.

[19] Tony M Rath and Rudrapatna Manmatha. Word spotting for historical documents. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2-4):139–152, 2007.

[20] Nicholas R Howe, Shaolei Feng, and R Manmatha. Finding words in alphabet soup: Inference on freeform character recognition for historical scripts. *Pattern Recognition*, 42(12):3338–3347, 2009.

[21] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer, 2007.

[22] Volkmar Frinken, Andreas Fischer, R Manmatha, and Horst Bunke. A novel word spotting method based on recurrent neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):211–224, 2012.

[23] Zhuoyao Zhong, Weishen Pan, Lianwen Jin, Harold Mouchère, and Christian Viard-Gaudin. Spottingnet: Learning the similarity of word images with convolutional neural network for word spotting in handwritten historical documents. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 295–300. IEEE, 2016.

[24] Sebastian Sudholt and Gernot A Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 277–282. IEEE, 2016.

[25] Raghavan Manmatha, Chengfeng Han, and Edward M Riseman. Word spotting: A new approach to indexing handwriting. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 631–637. IEEE, 1996.

[26] Arjun Sharma et al. Adapting off-the-shelf cnns for word spotting & recognition. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 986–990. IEEE, 2015.

[27] Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. A fast matching algorithm for graph-based handwriting recognition. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 194–203. Springer, 2013.

[28] Théodore Bluche, Hermann Ney, and Christopher Kermorvant. Feature extraction with convolutional neural networks for handwritten word recognition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 285–289. IEEE, 2013.

[29] François Chollet et al. Keras, 2015.

[30] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[31] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[32] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

Vita

A brief vita goes here.