INTERACTIVE MACHINE LEARNING FOR WORD SPOTTING ON DAMAGED HANDWRITTEN DOCUMENTS

By
Jack Bandy

Director of Project: Brent Seales

Director of Graduate Studies: Miroslaw Truszczynski

Date: April 19, 2018

MASTER'S PROJECT

Jack Bandy

The Graduate School
University of Kentucky
2018

INTERACTIVE MACHINE LEARNING FOR WORD SPOTTING 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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

This project deals with word spotting for historical documents, especially documents that have been physically damaged.

Extensive research exists in word spotting and word recognition on printed as well as handwritten documents. For modern data, the input to these problems is remarkably clean, but historical data, such as the data considered in this project, presents challenges that are not often considered in related literature. For example, although many projects for handwritten word recognition consider variations caused by penmanship, very few consider variations caused by physical damage.

This chapter reviews existing literature related to this problem, motivates an alternative approach, and outlines the contributions of the project.

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 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. As detailed in the following sections, the fields have converged at this point in time, but a distinction is necessary for the previous decades of work.

Text Recognition

From a technical standpoint, automatic text recognition is the task of turning an image into the text within the image. "Text recognition" here refers to recognizing *printed* texts, not handwritten texts, which prompts several convenient assumptions. Namely, one can assume that all occurrences of a given character will be identical in shape and size. Because of this assumption, researchers could attempt letter-for-letter recognition on documents, a process known as object character recognition (OCR).

OCR on scans of printed documents has seen success since as early as the 1980s [2, 3], with methods detailed as early as the 1950s [4]. A survey from 1996 [5] notes that, due to the consistency of letter shapes and sizes in question, simple techniques such as projection histograms, template matching, zoning, and geometric moments produced remarkable accuracy.

As early as 1987, font and size constraints were no longer needed. The authors of [6] demonstrated a system that accurately classified mixtures of dissimilar fonts of varied sizes. Gradually, more and more constraints were eliminated. After [6] 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," [7, 8, 9] with essentially no assumptions about the nature of the text. Especially important for "in the wild" recognition was eliminating the segmentation step, as in [10], 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 *without any additional input*.

An important benchmark dataset for text recognition "in the wild" is Street View Text (SVT) [11]. SVT was harvested using pictures from Google Street View, and thus contains a heterogeneous collection of word images with a variety of fonts, colors, and backgrounds. (Despite the variations, word images in this set do not include handwritten characters.) The SVT dataset was released in 2010, and by 2012, [8] used it to train a neural network that achieved state-of-the-art performance for both character recognition and word recognition. The high degree of accuracy was achieved via unsupervised feature learning and convolutional neural networks.

In fact, even before 2012, many researchers realized that convolutional neural networks (CNNs) were ideal for recognizing the shapes of different letters and words [12, 13], and the trend only became stronger after successes like [8]. CNNs offered exceptional performance with lower computational costs than traditional, "fully-connected" neural networks. Today, many robust approaches to text recognition exist via CNNs [8, 14, 9].

Handwriting Recognition

Although modern methods for printed text recognition overlap methods for handwriting recognition, especially with CNNs for "in-the-wild" handwriting recognition, the convergence happened after many years of parallel research, so it is helpful to examine both histories.

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 location details of the utensil help reveal the intended writing. For example, UNIPEN [15], 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 [16]. MNIST consists of grayscale images of individual handwritten digits, 0 to 9, and the objective is to classify each image by the digit written inside of

it. Machine learning researchers have been using this task as a benchmark for several decades [17], with error rates well below 1% since 2003 [18].

Projects using MNIST and similar datasets are premised upon many constraints, although they differ from those made for printed text. Rather than assuming consistency in size and shape, the projects assume a very small vocabulary or character set, which can be accurately recognized with proper alignment and segmentation. As soon as a text ventured outside those constraints (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) [19, 20, 21]. HMMs utilized statistical models built for specific languages to narrow down the classifications for a given letterform. A common example in english is that when a "q" occurs, a well-trained HMM will know to expect a "u" to follow. With HMMs, character and word recognition accuracies improved to over 85% (varying with respect to the test corpus) on "unconstrained" texts.

Although the texts were nominally unconstrained datasets, many demonstrations were still using the IAM dataset [22], 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 comprises hundreds of manuscript pages from the Library of Congress, handwritten by George Washington's secretaries. (A subset of this dataset is used in the evaluation portion of this project.)

In the mid-2000s, state-of-the-art HMM methods yielded word error rates around 50% on *truly* unrestricted 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. Retrieval systems allow users to query a dataset of images for a word, then scans the images for visual matches of that word. For example, [23] presents a retrieval system that achieves 63% mean average precision scores on the George Washington collection (this metric is discussed in section 2.5).

In [24], the word retrieval approach is formalized as a viable way to generate a searchable index of handwritten papers. Their method of "word spotting" turns the search problem into a clustering problem, where word images that are "closest" to the query word are considered matches. word spotting 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, rather than generating a full index beforehand, matching was now done in real-time.

Building upon the success of word spotting techniques and HMMs, [25] takes a step further and first detects handwritten *characters* in a word, then infers a word using an ensemble of HMMs. This approach allowed the recognition of words that were never seen during training, and established standards for character recognition within the George Washington dataset.

By the time ensemble HMMs came onto the scene, neural networks were already penetrating the field of handwriting recognition [26]. By 2010, techniques such as bidirectional long short-term memory (BLSTM) were successfully applied to word

spotting [11] and outperformed other methods. Finally, recurrent neural networks [27] 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 [28, 29]. 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 subsection, the scope of related projects is narrowed down from all handwriting recognition systems, and only research related to word spotting on historical documents is examined.

As previously mentioned, [24] formalized the idea of word spotting. However, the concept was originally proposed in [30], which clustered similar words to be annotated by users, and reported success for single-author documents with high-quality scans of the handwriting. [31] acknowledges some of the main challenges for historical documents: perfect line and word segmentation is nearly impossible, and unconstrained word recognition is extremely difficult. Although their proposed "transcript mapping" method assumes a pre-existing transcript for the input image, it introduces concepts for handling variable baseline position, line skew, character size, and inter-line distance.

A fairly complete system for retrieving words in handwritten manuscripts is presented in [23]. Once again, the solution steers away from the challenging task of full-on handwriting *recognition*, and instead focuses on retrieval of individual words. The goal is to provide a way for users to search the document, not necessarily to reveal its exact contents.

After performance improved on retrieval, around 2009, handwriting researchers returned to the recognition task. As mentioned in the previous subsection, a segmentation-free approach described in [25] utilizes character detection to improve state-of-the art performance on the George Washington manuscript. Because it is character-based, their system also recognizes words not seen in the training phase, the first such achievement for handwritten cursive manuscripts.

The Parzival database [32], a medieval German text from the 13th century, also became a popular evaluation benchmark, and [33] used it to demonstrate a system for full-on recognition, although it relied on a predetermined vocabulary. The following year, [10] matched state-of-the-art performance with a segmentation-free approach to word spotting. By 2016, just as they had done for printed text recognition and handwriting recognition, CNNs had set new standards for word spotting in historical documents [29, 28, 34] .

Finally, several projects have explored interactive approaches to word annotation, such as the one described in this paper. A familiar example is ReCAPTCHA [35], a system which methodically crops pictures of book pages into images of individual words, then, using many different users to label individual word images, it produces a

full transcript of the page and the book. Extending this method to take advantage of machine learning techniques, [36] demonstrates a similar system for generating ground truth for any dataset. Like ReCAPTCHA, it allows users to annotate manuscript excerpts with labels for characters and words. However, the goal of the system is to create training data for an ML-based word recognition or word spotting system.

1.2 Motivation

After the review of related work in printed text recognition, handwritten word recognition, and handwritten word spotting, they may appear to be solved problems. The explosion of machine learning research – in particular convolutional neural networks – has led to drastic improvements in performance on these tasks, and many advancements have even found their way to consumer products. For example, default software on most PCs 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 automatically transcribing damaged 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 presents the most accurate solution, it is incredibly time-consuming for larger documents. Moreover, on damaged documents, skilled papyrologists are required to decipher texts. This makes human transcription 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, as character-based algorithms are unable to handle incomplete words.

An ideal solution would leverage automated transcription for the undamaged portions, and allow a human reader to fill in any gaps. Figure **??** shows the potential architecture of such a framework, which was presented in [1]. This architecture can be referred to as semi-automated transcription, and this paper details a pipeline for semi-automated transcription, blending the irreplicable abilities of the human eye with the efficiency and scalability of character recognition algorithms.

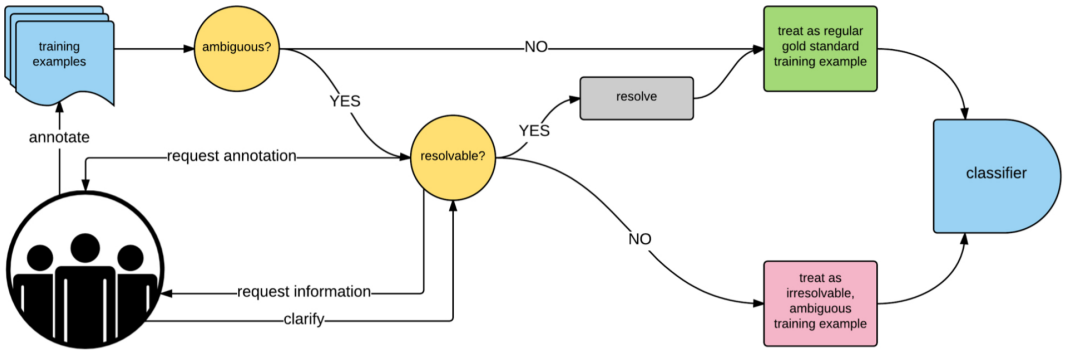

Figure 1.1: A schematic diagram from [1] for semi-automated transcription, designed especially for resolving ambiguity. Although resolving ambiguity is not the focus of my project, this diagram depicts a collaborative framework between machine learning and human annotation.

1.3 Contributions

The methods used in this project borrow heavily from methods in the aforementioned research areas, especially keyword spotting [37, 27], word recognition [25], and handwriting recognition [38, 39].

Nonetheless, the 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 leverages user-provided labels of words in a given document. Essentially, the system allows users to label an image, and then uses that label to annotate similar word images. This essentially lets the user label a cluster of images, while still allowing a user to correct an incorrect classification from the clustering algorithm.

A Technique for Virtual Ink Restoration

To deal with the problem of damaged and distorted words, this project presents a convolutional autoencoder (CAE) for restoring words with missing or incomplete letters, which is successfully trained to capture and recreate the original appearance of a damaged word. Not only does the recreation lead to accuracy improvements for automated recognition, it also allows users to view an enhanced version of the damaged manuscript.

Chapter 2 Methodology

2.1 Preprocessing

For two of the datasets used to evaluate the system (the George Washington and Parzival datasets), segmented and binarized word images are already provided courtesy of [40]. 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 cropping would take place in the line segmentation and word segmentation algorithms. The key in column cropping is to create images that are vertically aligned and 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, the margin 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. Next, the image is thresholded to create a binarized representation. Because lighting and coloring varies across manuscript pages, a simple global threshold would lead to noisy and inconsistent background removal. Instead, a threshold should be calculated individually for each page – a process referred to as "adaptive thresholding." Considering the nature of the data leads to a natural choice of algorithm for this process.

Because the column image is assumed to only contain ink and paper, a histogram of values in the column image should be bimodal (with the two peaks representing the approximate value of an ink pixel and a paper pixel). Otsu's binarization algorithm [41] 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. The system uses Otsu's thresholding algorithm as implemented by OpenCV [42].

Line Segmentation

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

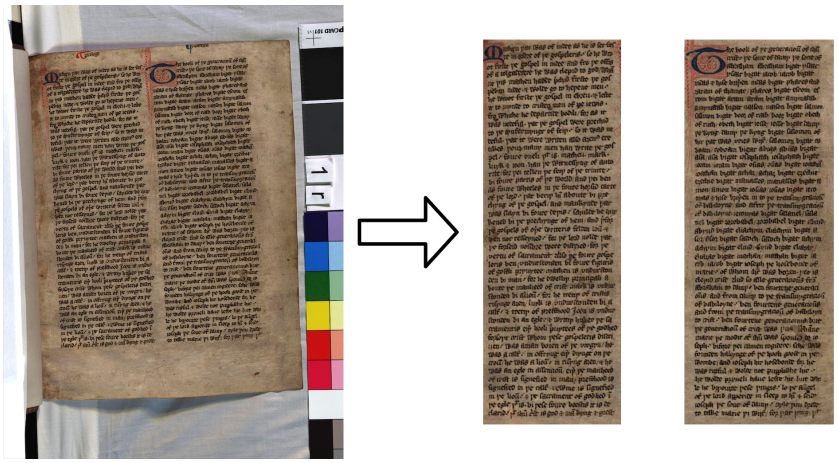

Figure 2.1: A sample of the original photographs of the Wycliffe New Testament Manuscript. In the preprocessing phase, these images must be aligned and cropped into separate columns.

The Wycliffe New Testament is aligned and spaced with remarkable consistency, and the segmentation technique takes advantage of this. A vertical projection profile of a binarized page image is used to determine the approximate location of individual lines of text, because the relative minimum values of the profile correspond to the spaces in between lines of text. This is visualized in Figure 2.2.

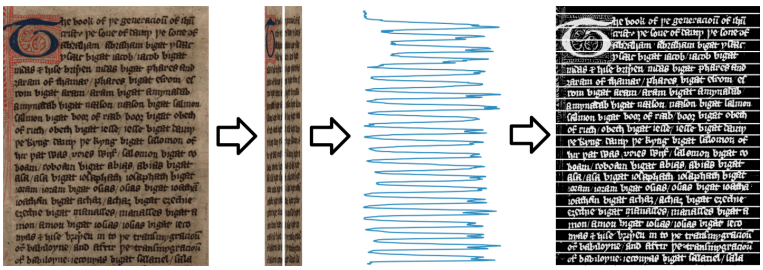

Figure 2.2: The projection profile used to segment lines of text. To generate words, an identical process

However, because lines are relatively wide (10-12 words), writing on some pages is slightly slanted. Figure 2.3 illustrates the dilemma of horizontal segmentations.

To segment tilted lines of text, a line of best fit is generated across the words in a given line. This is achieved by taking the coordinates of all the nonzero values (i.e. values above the ink threshold) in the approximated horizontal text region, and applying random sample consensus (RANSAC) [43] to generate a line that fits the overall tilt of the given line. The resulting approximation eliminates most word cutoff.

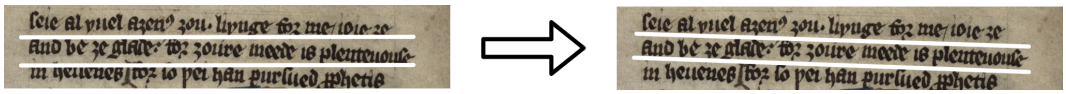

Figure 2.3: Illustrating the need for tilt during the line segmentation phase. Although the column has been aligned vertically, perfectly horizontal approximations (above) for line segmentation result in cutoff words. Tilted lines generated by the RANSAC algorithm fit the slant of the text.

Word Segmentation

After a line is segmented, the line must be divided into individual words. A similar process to line segmentation is used. Once a horizontal projection profile of a text line is generated, the system applies a gaussian filter to eliminate noise in the dataset. This helps ensure that local minimums in the profile correspond to word gaps.

Once the system finds local minimums in the profile, it checks whether their values dip below a threshold. This threshold was calculated as follows:

$$mean(P) - \frac{stdev(P)}{2}$$

where P is the horizontal projection profile of the text line.

Damage Simulation

The George Washington and Parzival datasets exist in fairly high quality. So, to demonstrate the system's ability to handle damaged datasets, simulated damage was performed on the word images. A simple approach was used, and works as follows.

One half of the images in the dataset are chosen at random to be "damaged." Each of these images is treated with a random number of damage blocks, at least two and at most six. Each damage block is a rectangle of random height and width, from sixteen pixels to thirty pixels. The damage block is centered at a random nonzero pixel in the image, and each pixel contained in the block is set to zero. An example of damage simulation is shown in figure 2.4.

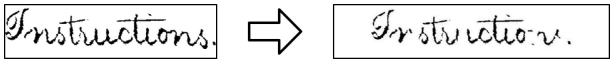

Figure 2.4: An original word image from the George Washington dataset (left), and the same word after applying damage blocks to simulate ink deterioration (right).

Baseline Feature Extraction

The histogram of oriented gradients (HoG) feature was used as a baseline feature for the images. Although more advanced features are available that result in better

accuracy, the goal was to apply a proven technique that would provide adequate baseline performance. This portion of the system is modular, so an improved feature extraction system would be straightforward to implement.

HoG features were extracted using a scikit-image implementation [44]. The results in this paper were achieved using a HoG descriptor with 9 orientation bins, a cell size of 16pixels x 16pixels, 2x2 cells per block, and L1 normalization.

2.2 Convolutional Autoencoder

The purpose of the Convolutional Autoencoder (CAE) [45] in this system is to learn an encoded representation of word images, without the need for ground truth label. After being trained, the CAE can be used to encode word images into features, as well as to enhance damaged words before word spotting takes place.

The CAE is an unsupervised generative technique for learning a latent representation of input data. In this case, the autoencoder works on top of a convolutional neural network and is trained using stochastic gradient descent, where the loss function is the cross-entropy between the output image and the input image. As a generative model, its job "is to somehow capture the dependencies between pixels" [46]. After proper training, the CAE should be able to take an input image, create an encoded representation of that image, and then generate a similar image using only the encoding.

For example, an idealized CAE trained on the MNIST dataset would learn an encoding that captured the digits 0 to 9. This encoded layer of the network is referred to as the latent variable, and the latent variables are used to decide which class of output should be generated. If latent variables corresponded to a 1, the decoding portion of the network would generate output that resembled a 1.

The CAE trains on each dataset separately, so that separate models are created for the George Washington dataset, the Parzival dataset, and the Wycliffe dataset. In the evaluation chapter, these models generate "restored" versions of the word images before word retrieval takes place, and results are compared to the non-restored word image set.

The CAE was implemented using Keras [47], with TensorFlow [48] used as the backend. The full architecture, visualized in figure 2.5, includes only three kinds of layers: convolutional layers, pooling layers, and upsampling layers.

CAE for Feature Extraction

When using the CAE as a feature extractor, no changes are made to the network architecture: the network receives an input image and performs the convolutional filters just the same as a normal prediction. The only difference is that the network is pruned at the final convolutional layer, or the "encoded representation" layer. The output of this layer is flattened into a 1-dimensional feature vector to be used in word spotting.

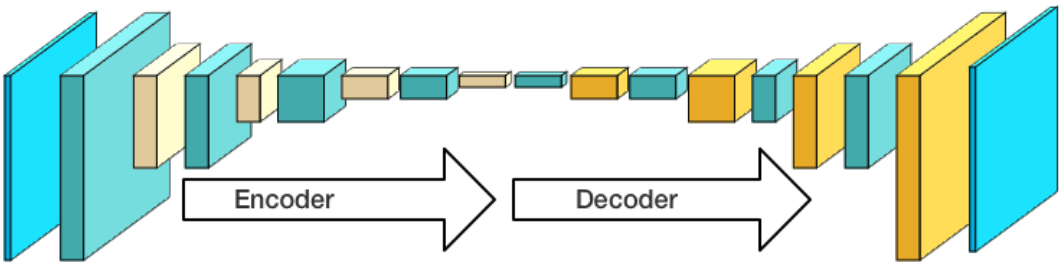

Figure 2.5: A diagram of the convolutional autoencoder. The thin blue layers at the far left and far right represent the input and output, respectively. Teal blocks represent convolutional filters, white blocks represent pooling layers, and yellow blocks represent upsampling layers.

CAE for Restoration

For damaged datasets, it is desirable to reconstruct the input to more closely resemble their original form. In the case of damaged word images, restoration involves replacing blank or missing pixels with expected values in a way that completes the shape of the expected word. The computer vision community refers to this task of filling in corrupted images as "inpainting."

CAEs have previously been used for inpainting [49], and can be trained to do so with a fairly simple modification: rather than using the same image as input and ground truth, a damaged version of the ground truth is used as the input image. Then, training for the CAE involves learning how to create the undamaged version of the image based on the input/damaged version. This method is used to train a network hereafter referred to as the restorative network, and results can be seen in figure 3.3.

2.3 Word Retrieval

The current system employs a simple query-by-example method. This means that the query requires a word image as input, rather than a string. Given a word image as input, the query system calculates the feature vector (either HoG or CAE encoding) and compares it to the feature vectors for all other word images. The results are sorted by the cosine distance between vectors, and returned in order of proximity.

2.4 Providing Labels

Users can provide labels to word images 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, flag the word image as requiring re-segmentation, or search for other occurrences of the word. The label is stored in a simple file that also records the coordinates of the word and whether segmentation was affirmed.

Priority Queue

The order in which to request labels from users is a key concern. Two simple choices would be to present words in order of appearance, or to present them randomly.

A more useful approach, leveraged by this system, allows users to label "interesting" words. In some cases, that could mean the most frequently occurring words, and in other cases, it could mean labeling the rare words. The general approach remains the same: after features are extracted from word images, agglomerative clustering is performed on the feature vectors, using the estimated number of unique words in the data to determine how many clusters should be formed.

Once word images are clustered, labeling common words is a matter of labeling words in the largest cluster. Words that occur only once (*hapax legomenons*) correspond to clusters of size one. The system allows users to label words randomly, in order of appearance, in order of frequency (common words first), or in reverse order of frequency (rare words first).

2.5 Evaluation

Unique metrics are used to objectively evaluate word spotting methods. Most of these metrics are primarily concerned with the *precision* of the results. In other words, when a word is queried and the system provides a list of matching word images, the matches are evaluated based on whether those results are correct, and not based on whether it missed other matches in the data.

Precision at K

The precision at k ($P@k$) metric is one of the most widely used for evaluating word spotting methods. $P@k$ determines the precision for the k top retrieved words, or the relevance of the k top results.

$$P@k = \frac{\mid \{relevant\ instances\} \cap \{k\ retrieved\ instances\} \mid}{\mid \{k\ retrieved\ instances\} \mid}$$

So if k=5, and a query for the word "october" returned three instances of "october" and two instances of "octopus," the $P@5$ score score would be $\frac{3}{5} = 0.6$ for that query. This project uses $P@5$, as do most other word spotting papers from the past decade, reviewed in [50].

Mean Average Precision

The mean average precision (MAP) is an effective metric to holistically evaluate any word spotting system. A recent review [50] of nearly 200 papers in word spotting tracked the evaluation metric used in each paper, and conclusively found MAP to be dominant. The formal definition for *Average Precision* is as follows:

$$AP = \frac{\sum_{k=1}^{n}(P@k\ \times rel(k))}{\mid \{relevant\ instances\} \mid}$$

MAP is simply the mean value of AP over all queries. The metric rewards systems that properly sort results according to relevance. If two systems each have three relevant results in the top five matches, they can still produce different scores. For example, if one system puts the relevant results in the top three slots while the other fails to do so, the former system will produce a better MAP score.

Chapter 3 Results

The evaluation phase sought to determine whether CAE encodings of word images could be used for word spotting, and also if a CAE could reconstruct damaged words so as to more accurately classify them.

3.1 Datasets

Three datasets were used to evaluate the CAE-based approach to word recognition and ink restoration. The first two are benchmark datasets from the literature, and the third is a custom scan of a publicly available document. Samples of all three datasets are shown in figure 3.1.

Dataset	Year	Medium	Words	Unique Words
George Washington	1755	Ink on paper	4,894	1,471
Parzival	1200s	Ink on parchment	23,478	4,934
Wycliffe*	1388	Ink on parchment	310	143

Table 3.1: Summary table of the datasets used for evaluation. Note the Wycliffe test set is a small subset of the full Wycliffe New Testament.

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 [40], in which authors provide word segmentation, ground truth, and normalized images for 20 pages of the George Washington letters.

Parzival Dataset

The Parzival database is a medieval German text from the 13th century which was annotated and made publicly available by [32]. It includes nearly 50 manuscript pages. The ink and parchment closely resemble the Wycliffe documents, which is not too surprising given their chronological proximity.

Wycliffe Dataset

The Wycliffe New Testament is a Middle English Bible translation from the 14th century. Numerous copies of Wycliffe's New Testament survive to this day, but the particular manuscript images used in this project were acquired in 2010 from...?

Challenges discussed in section 4.2 explain why this test set was relatively small compared to the full Wycliffe New Testament. However, the system does allow query-by-example over the full manuscript.

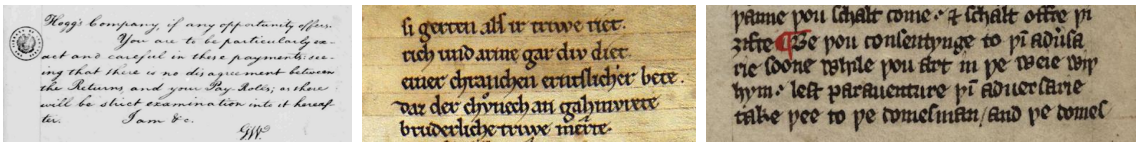

Figure 3.1: Sample lines from the George Washington dataset (left), the Parzival dataset (middle), and the Wycliffe dataset (right).

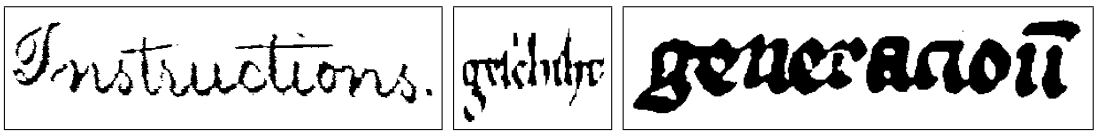

Figure 3.2: Sample word images from the George Washington dataset (left), the Parzival dataset (middle), and the Wycliffe dataset (right).

3.2 Basic Word Spotting

The goal of the basic word spoting experiments was to determine whether the latent variables in a CAE could adequately serve as features during word spotting. Histogram of oriented gradients (HoG) is a popular feature extraction method for word spotting [50]. Although other optimizations combined with advanced feature extraction produce better results, HoG is a sufficient benchmark with which to compare alternate features.

Table 3.2 shows that CAE-encoded features outperform HoG features on all three datasets. Word spotting precision at k=5 (P@5) and mean average precision (MAP) over all queries are shown in the table. Although CAE features offer negligible improvement on the George Washington dataset, it is the decisive winner for the Parzival and Wycliffe sets.

Data	**Feature**	**P@5**	**MAP**
Original GW	HoG	0.747	0.675
Original GW	CAE Encoding	0.748	0.678
Original Parzival	HoG	0.689	0.637
Original Parzival	CAE Encoding	0.771	0.720
Original Wycliffe	HoG	0.631	0.584
Original Wycliffe	CAE Encoding	0.768	0.662

Table 3.2: Word Spotting Results

3.3 Classification on Damaged Datasest

The goal of experiments on damaged word images was twofold: first, to determine whether CAE features were more suitable for damaged data, and second, to set a

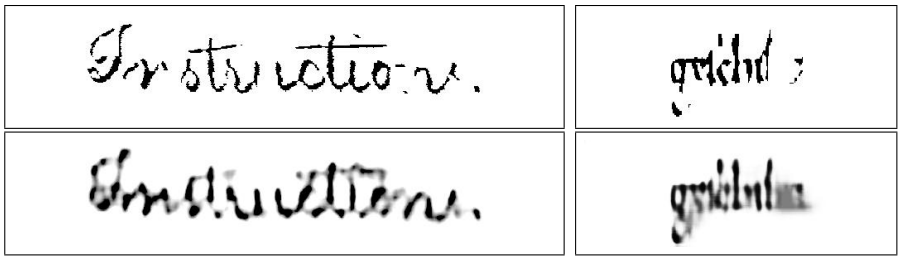

Figure 3.3: On the top, samples of word images after simulated damage, from the George Washington dataset (left) and the Parzival dataset (right). Respective output from the reconstructive CAE is shown on the bottom.

baseline performance to be compared against in the following section.

Table 3.3 shows that CAE encoding outperforms HoG on the damaged datasets, even on the George Washington dataset which saw little difference between features in the previous section.

Data	**Feature**	**P@5**	**MAP**
Damaged GW	HoG	0.602	0.547
Damaged GW	CAE Encoding	0.650	0.592
Damaged Parzival	HoG	0.515	0.477
Damaged Parzival	CAE Encoding	0.615	0.572

Table 3.3: Results for "Damaged" Datasets

3.4 Reconstruction Results

Finally, word spotting performance is evaluated on reconstructed word images. The change represents the improvement over classification results on the original data (Wycliffe) or damaged data (George Washington and Parzival). Details on the restorative CAE can be found in section 2.2, and discussion of these results can be found in section 4.1.

Data	**Feature**	**P@5 (change)**	**MAP (change)**
Reconstructed GW	HoG	0.644 (+0.042)	0.583 (+0.036)
Reconstructed GW	CAE Encoding	0.665 (+0.015)	0.604 (+0.012)
Reconstructed Parzival	HoG	0.554 (+0.039)	0.512 (+0.035)
Reconstructed Parzival	CAE Encoding	0.620 (+0.005)	0.577 (+0.005)
Reconstructed Wycliffe	HoG	0.873 (+0.242)	0.768 (+0.184)
Reconstructed Wycliffe	CAE Encoding	0.859 (+0.091)	0.755 (+0.093)

Table 3.4: Results for Reconstructed Data

Chapter 4 Conclusion

4.1 Findings

Besides demonstrating a semi-supervised approach to word spotting in historical documents, this project demonstrated that CAEs are useful at two points in the word spotting pipeline. First, the encoded representation of word images can be used as features for the matching process. Second, generative CAEs can be used to restore damaged word images, producing an enhanced version for subsequent word spotting.

CAE for Feature Extraction

Results on three separate datasets showed conclusively that the encoded representation of word images outperformed HoG features for word spotting. This was the case in both section 3.2 and 3.3, where the *P@5* and *MAP* metrics were higher when matching CAE features.

Results on the damaged datasets, shown in section 3.3, were especially clear. The CAE features prove to be more robust to simulated damage, likely because the latent variables capture high-level characteristics. If a small portion of the word image is changed, the CAE is still able to encode those characteristics to some degree, and the resulting feature vector will still reflect some similarity.

In contrast, HoG features change immediately after changes to a single pixel value. Thus, the damage regions created a significantly different feature vector that fails to capture the higher-level characteristics, leading to worse performance on the damaged datasets.

CAE for Ink Restoration

A restorative CAE was shown to improve results on datasets with simulated damage, as well as the Wycliffe test set. The example reconstructions shown in figure 3.3 suggest that the network reconstructs images well enough to be visually recognizable to a human, and results in 3.4 demonstrate that word spotting results improve significantly on the reconstructed data.

This finding is immediately useful for word spotting systems that work on damaged handwritten documents. The CAE can be understood as an enhancement to the raw word images which lead to more useful feature extraction for word spotting. Not only does the network "fill in" damaged portions of the word image with plausible pixel values, it also causes different occurrences of the same word to more closely resemble each other. This leads to more similar feature vectors between word occurrences, and greater word spotting accuracy as the end result.

4.2 Challenges and Limitations

Although the word spotting system detailed in this paper demonstrably improves recognition on damaged datasets, various challenges arose at certain parts of the pipeline. Precise segmentation, ground truth collection, and query-by-example present three challenges encountered which limit the final product.

Although segmentation-free approaches to word spotting have been gaining popularity, this project assumed word segmentation. Specifically, the CAE assumed input samples of a fixed dimension, and that each input sample would contain a single word. For the George Washington and Parzival datasets, the segmentation phase was already completed, but for the custom Wycliffe dataset this was not the case.

The methods section details an algorithm which accurately finds lines of words. After fine-tuning the thresholds and introducing gaussian smoothing to the projection profile, this algorithm performed remarkably well on every page. However, the same adjustments did not produce reliably accurate word segmentation. The space between words varies enough that, unlike line segmentation, a global threshold is unsuitable. Thus, the resulting word images for the Wycliffe dataset often include more than one word (when words were close together) or a partial word (when letters in a word were far apart). This led to the "segmentation error" button in the GUI, which flags the sample for re-segmentation.

Because automatic word segmentation could not be trusted, it proved impossible to generate ground truth for the Wycliffe dataset. Once the segmentation system made a single error, each subsequent word in the transcript would be mapped to the incorrect word image. To evaluate the system, manual segmentation of words was performed on the first page of the manuscript to ensure that each word image corresponded to the correct ground truth label.

Lastly, the semi-supervised nature of the system means query-by-text is unsupported until at least one instance of the text has been labeled. The user can still choose to search for matching word images of any word image encountered in the labeling interface. This search, however, is a query-by-example approach, which is less useful for textual analysis.

4.3 Future Work

Alternate Generative Models

Those exploring the use of CAEs for word spotting should investigate alternative approaches to generative neural networks. The CAE relies on fairly simple statistical models which can limit their performance. Generative Adversarial Networks (GANs) [51] leverage a game-like training approach, while autoregressive models [52] aim to learn a conditional distribution of the data. Although these different generative approaches present their own unique challenges, compromising on attributes such as training time and efficiency can lead to better results in the long-term.

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] MH Glauberman. Character recognition for business machines. *Electronics*, 29(2):132–136, 1956.

[5] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4):641–662, 1996.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] Zohra Saidane and Christophe Garcia. Automatic scene text recognition using a convolutional neural network. In *Workshop on Camera-Based Document Analysis and Recognition*, volume 1, 2007.

[13] Manolis Delakis and Christophe Garcia. text detection with convolutional neural networks. In *VISAPP (2)*, pages 290–294, 2008.

[14] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):970–983, 2014.

[15] 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.

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] Catalin I Tomai, Bin Zhang, and Venu Govindaraju. Transcript mapping for historic handwritten document images. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 413–418. IEEE, 2002.

[32] Andreas Fischer, Emanuel Indermühle, Horst Bunke, Gabriel Viehhauser, and Michael Stolz. Ground truth creation for handwriting recognition in historical documents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 3–10. ACM, 2010.

[33] Andreas Fischer, Micheal Baechler, Angelika Garz, Marcus Liwicki, and Rolf Ingold. A combined system for text line extraction and handwriting recognition in historical documents. In *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*, pages 71–75. IEEE, 2014.

[34] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Deep feature embedding for accurate recognition and retrieval of handwritten text. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 289–294. IEEE, 2016.

[35] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[36] Ofer Biller, Abedelkadir Asi, Klara Kedem, Jihad El-Sana, and Itshak Dinstein. Webgt: An interactive web-based system for historical document ground truth generation. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 305–308. IEEE, 2013.

[37] 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.

[38] 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.

[39] 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.

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

[41] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[42] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb?s journal of software tools*, 3, 2000.

[43] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987.

[44] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[45] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[46] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

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

[48] 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.

[49] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[50] Angelos P Giotis, Giorgos Sfikas, Basilis Gatos, and Christophoros Nikou. A survey of document image word spotting techniques. *Pattern Recognition*, 68:310–332, 2017.

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[52] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Vita

A brief vita goes here.