A Neural Network Approach to Handwriting Recognition in the Codex A

by

Eleanor Anthony

A thesis submitted to the faculty of The University of Mississippi in partial fulfillment for the requirements of the Sally McDonnell Barksdale Honors College.

Oxford
May 2016

Approved By:

Advisor: Dr. Gregory Heyworth

Reader: Dr. Dawn Wilkins

Reader: Dr. Yixin Chen
“As civilized human beings, we are the inheritors, neither of an inquiry about ourselves and the world, nor of an accumulating body of information, but of a conversation, begun in the primeval forests and extended and made more articulate in the course of centuries. It is a conversation which goes on both in public and within each of ourselves.”

Michael Oakeshott, ”The Voice of Poetry in the Conversation of Mankind”
Abstract

In this research, an attempt to create a neural network classifier for the Latin uncial alphabet found in the Codex A, a late fourth century manuscript held by the Archivo Capitolare in Vercelli, Italy and the oldest of the Latin Gospels, is outlined. Training data collection methods, as well as image processing and artificial data creation methods used for expanding the training set, are discussed, as is their contribution to the success of the neural networks. While successful results are achieved for the testing sets, classification of new image data is largely unsuccessful. Potential improvement and avenues for expanding the research are discussed, as are potential explanations for the results.
Acknowledgements

First, I’d like to thank my thesis advisor and research director Dr. Gregory Heyworth. His relentless encouragement, support, and guidance were invaluable in the completion of this thesis, and I am so grateful for all the experiences and memories I have gained working with the Lazarus Project research team. I would also like to thank Mr. Ken Boydston and Dr. Roger Easton for sharing their expertise and including research students like myself in imaging projects. The Lazarus Project researchers have done so much to foster the growth of young scholars, and I am thankful to be among them.

I would also like to thank Dr. Jane Xue for her initial advisement of this thesis. While we were unable to continue working together after a professional move, her guidance shaped the nature of my research and my understanding of it, for which I am extremely grateful. I wish her the best as she pursues a career at Google!

Many thanks to my readers, Dr. Dawn Wilkins and Dr. Yixin Chen. The computer science department has been so supportive of my research, even stepping in to help with computer issues at the last minute when I desperately needed it! I so appreciate your time investment in reviewing my project and your patience with its development.

Additional thanks to Mr. Jeff Lucas for all his help in securing a computer to complete this research! I would also like to thank Mr. Ben Pharr for his help in attempting to operate EBLearn on the UM supercomputer and for enduring the ensuing headache.

I so appreciate the support that the Sally McDonnell Barksdale Honors College has provided specifically for this research, and for their support of my development as a student more generally. Special thanks to Dr. Douglass Sullivan-Gonzalez, Dr. Debra Young, Dr. John Samonds, Mr. Bruce Levingston, Mr. Fred Carraway, and Ms. Penny Leeton for all their support in this endeavor.

I would also like to thank the generous gift of the Carrier family that has funded my education at the University of Mississippi. I have enjoyed a wonderful five years at this school, and I am excited about moving into the next phase of my education having completed a challenging and fulfilling undergraduate degree at this university.
Finally, many thanks to my friends and family for supporting me in this process. I couldn’t have completed this research without the constant questioning, late night writing sessions, and ice cream runs that you all provided. Thank you all for your encouragement!
List of Figures

1.1 Basic Neural Network .............................................. 3
2.1 Matlab Training Data Collection Tool ........................... 8
2.2 Latin Alphabet in Uncial Script ................................. 8
2.3 Training Data: Original, Quantized, and PCA Images ........ 11
2.4 Manually Altered Training Data ................................. 12
3.1 Lenet 5 Architecture ............................................... 15
4.1 V1 Testing Set Class Examples ................................. 17
4.2 Cartoon Net: Training and V1 Success Rates ................. 17
4.3 Detect Function Results for BG PCA+Cartoon+Shift Net .... 21
# List of Tables

2.1 Number of Original Images in Training and Testing Sets for Character Classes ..................................................... 9

2.2 Number of Training Images Produced in Artificial Training Production ................................................................. 11

4.1 Number of Images in Training and V2 Testing Set for BG PCA+Cartoon+Shift ....................................................... 19

4.2 Success Rates for Nets Selected for Highest V1 Recognition Rates .............................................................. 20

4.3 Success Rates for Nets Trained with Background Images Selected for Highest V1 Recognition Rates .................. 20

4.4 Success of Character Identification in New Manuscript Image Data ........................................................................ 22
Chapter 1

Introduction

The human ability to discern patterns is remarkably robust and evolutionarily advantageous. Our ability to perceive objects, sounds, and sights and classify them based on their various characteristics is critical to functioning in our world. However, the human ability for classification has limitations; noise, clutter, speed, and intensity are all factors that affect our ability to recognize patterns.

Signal classification has thus become the subject of intense interest for computer scientists and mathematicians alike. Researchers have developed computational approaches to pattern recognition aimed at addressing the problems that face human recognition capabilities. In some cases, researchers have also sought to mimic human classification mechanisms.

Handwriting recognition in particular has been the subject of much research [1] [2]. It has proved a problematic classification task because of the irregularity present within the character classes of an alphabet. Clearly, different individuals write characters differently, though the appearance of characters written by single individuals also exhibit a great degree of variability.

This variability makes it very difficult to classify characters based on hand-designed heuristics of the data. Fortunately, an automatic learning method has been developed that nicely accommodates this variability [3]. Neural networks, originally designed to resemble the human brain’s network of neurons [4], extract features from character data, though they determine which features are relevant to classification, and to what degree, using training data. The variability present within this data affects the weight placed on any particular feature, making the nets ability to classify new characters more robust than that of classifiers using hand-crafted features.
This thesis will mostly focus on the use of a neural network to address the problem of handwriting recognition for the Codex A. Because the Codex A has suffered damage and decay due to age and water exposure, the text is extremely degraded in large portions of the manuscript. This introduces a new difficulty for human classification of characters in the manuscript, incentivizing us to find a computational approach.

1.1 The Codex A

The Codex Vercellensis Evangeliorum, more commonly known as the Codex A, is a late fourth century manuscript held by the Archivo Capitolare in Vercelli, Italy. The oldest of the Latin Gospels, this manuscript has been an object of cultural, literary, and linguistic interest to scholars and religious practitioners alike for hundreds of years. Unfortunately, recent research has been forced to consider the manuscript in a damaged form. Water damage and time have faded the text in large portions of the manuscript, while some portions of pages are also missing.

There have been attempts at providing a clearer understanding of the damaged text in the Codex A using multispectral technology. Scholars and researchers with the Lazarus Project, a research initiative directed by Gregory Heyworth of the University of Mississippi, traveled to Vercelli in July of 2014 to photograph the manuscript with 30 different wavelengths of light, ranging from the ultraviolet to the infrared, with transmissive and overhead light sources. Boasting successful projects like the revelation of the undertext of the Archimedes Palimpsest, multispectral technology and particular image processing techniques can make visible textual elements that time and damage have erased from view. However, while multispectral technology can make textual elements visually clearer, it often does not provide conclusive results, especially in the cases of partial allographs. Herein lies the motivation for implementing a computational approach for the recognition of the text.

1.2 EBLearn

The problem of recognizing handwriting is one of both academic and commercial interest. While the need for mail and check processing systems has created a market for its solution, considerable research has also been dedicated to developing open-source methods.
EBLearn represents the most developed open-source resource available to neural network developers. EBLearn is an open source library of C++ machine learning algorithms. Developed by Pierre Sermanet and Yann LeCun in New York University’s machine learning lab, it also utilizes optimizations, tools and cross-platform support added by Soumith Chintala. In particular, EBLearn supports the training and development of convolutional neural networks using energy based models. EBLearn also provides demos and tutorials for ease of use [5].

1.3 Neural Networks

Neural networks are biologically-inspired configurations of "neurons" which connect to one another in much the same way that the neurons in a human brain might. Neurons can send signals to one another based on the input that they are receiving, and these signals allow the net to recognize familiar pieces of data [4].

1.3.1 Net Topology

A basic neural network is shown in Figure 1.1. In this net, there are three layers: an input layer, a hidden layer, and an output layer. The hidden and output layers are made up of feature maps: the hidden layer is made up of five, and the output layer, two. These feature maps are produced by applying a function across sub-regions of the input images, adding a bias term, and then applying a non-linear function. If we denote the $n$-th feature map at a given layer as $h^n$, whose filters are determined by the weights $W^n$ and bias $b_n$, then the feature map $h^k$ is obtained
as follows (for \( \text{tanh} \) non-linearities) [6]:

\[
h_{ij}^n = \text{tanh}((W^n * x)_{ij} + b_n) \quad (1.1)
\]

### 1.3.2 Convolution

The \(*\) operation given in the equation above denotes a convolution operation. The convolution operation for a discrete set, such as the pixels in an image matrix, enables one to multiply a kernel throughout an input image and then sum the individual products. This number then becomes the value in the output matrix. The equation for the two dimensional convolution operation on a discrete set is given by [6]

\[
x[m, n] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} y[i, j] \delta[m - i, n - j] \quad (1.2)
\]

where \( x \) is the output matrix, \( y \) is the input matrix, and \( \delta \) is the convolutional kernel. It should be noted that this kernel’s dimensions must not exceed the dimensions of the input matrix. The size of the output matrix is equal to the difference of the dimensions of the input matrix and the kernel. For example, if \( x \) is a 5x5 matrix and \( \delta \) is 2x2, \( y \) will be 3x3.

### 1.3.3 Sigmoid Function and Bias Term

Returning to equation 1.1, we see that the sigmoid function is \( \text{tanh} \); this function ensures that the sum \( h_{ij}^n \) associated with each neuron is continuous and thus differentiable, a condition necessary for use of the learning algorithm by the net. Moreover, the sigmoid function normalizes the output for each neuron [7].

Biases are values that are added to the sums calculated at each neuron (except input neurons), given by equation 1.1. Their use in a neural net increases the capacity of the network to solve problems by offsetting the hyperplanes that separate individual classes for superior positioning. Weights, on the other hand, are used to determine what contribution each neuron should make to classification. They are first chosen randomly by the neural net and then through the employment of the backpropagation algorithm, they are altered to produce the desired output [7].
1.3.4 Backpropagation Algorithm

The backpropagation algorithm is used to determine and alter the net’s weights based on the error the net produces in calculating the known outputs. Each iteration of training the net proceeds as follows: 1) an instance of training data is fed through the network in a forward direction, 2) error is calculated at the output neurons based on known target information, 3) changes to the weights that lead into the output layer are determined based upon this error calculation, and 4) changes to the weights that lead to the preceding network layers are determined based on the properties of the neurons to which they connect[7].

The backpropagation algorithm uses as a performance index the mean square error between the input and desired output. The standard formula for square mean error is given by

\[ E = (t - y)^2 \]  

(1.3)

where \( E \) is the error, \( t \) is the expected output, and \( y \) is the actual output. The backpropagation algorithm aims to find the weights that minimize this error using gradient descent, such that the partial derivatives of the error are calculated with respect to the weights. Formalized, the change in weight calculation is given by equation 1.3 and the \( \delta_{jp} \) for output neurons and intermediary neurons is given by equation 1.4,

\[ \Delta w_{ijm} = \epsilon \delta_{jp} a_{iq} \]  

(1.4)

\[ \delta_{jp} = \begin{cases} a_{jp}(1 - a_{jp})(t_{jp} - a_{jp}), & \text{if output neuron} \\ a_{jp}(1 - a_{jp}) \sum_{x=0}^{n} \delta_{kx} w_{jkx}, & \text{if intermediary neuron} \end{cases} \]  

(1.5)

where node \( p \) is the node to which the vector associated with weight \( m \) leads, node \( q \) is the node from which the vector associated with weight \( m \) leads, \( i \) is the emitting layer of nodes, \( j \) is the receiving layer of nodes, \( k \) is the layer of nodes that follows \( j \), \( ij \) is the layer of weights between node layers \( i \) and \( j \), \( jk \) is the layer of weights between node layers \( j \) and \( k \), \( w \) are weights, \( a \) are node activations, subscripts refer to particular layers of nodes \((i,j,k)\) or weights \((ij,jk)\), sub-subscripts refer to individual weights and nodes in their respective layers, and \( \epsilon \) is the learning rate [7].
1.3.5 Success of Neural Network Application

Neural networks have been successfully implemented for a number of recognition tasks, including handwriting recognition, the detection of objects [8] and the detection of printed text [9]. In 1989, Yann Lecun and his fellow researchers were the first to achieve state-of-the-art recognition results utilizing a neural network. They trained a net to classify zip codes, achieving an error rate of 0.14% for their training set (10 mistakes) and an error rate of 0.5% for their testing set (102 mistakes) [10]. Lecun and his team went on to achieve similarly brilliant results on the SVHN dataset (a dataset of images of house numbers), achieving a recognition rate of 95.10%, 4.5 points of improvement over the previous state-of-the-art. [11]. Over time, this recognition rate has been successfully improved upon by numerous researchers, and the current lowest error rate has been set by Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu at 1.69% with the use of deep neural networks. [12].

Lecun has also compiled and released a dataset of handwritten digits that has received a great deal of attention from researchers in the field. For over a decade, researchers have tried to achieve the best error rate for the set, and the current titleholders are Dan Ciresan, Ueli Meier, and Jurgen Schmidhuber, achieving an error rate of 0.23% by utilizing a hierarchical system of neural networks [13]. Previous attempts have been made utilizing a K nearest-neighbors approach [14] and support vector machines [15]. Significantly, however, the top four lowest classification error rates have been achieved by researchers implementing neural networks [13] [16] [17].

In the same paper where they present their impressive MNIST results, Dan Ciresan, Ueli Meier, and Jurgen Schmidhuber prove that their multi-column deep neural networks can detect objects with an accuracy that surpasses human abilities. They achieve an error rate of 0.54% for the GTSRB traffic sign dataset. The human error rate for the set is twice that [13]. These results, and numerous others, show the effectiveness of neural networks as foundations for recognition systems and partially motivate the use of neural networks for the problem at hand: handwriting recognition in the Codex A.
Chapter 2

Training Data

It has been shown both theoretically and experimentally that the gap that exists between the expected error rate of the testing set, $E_{test}$, and the error rate of the training set $E_{train}$, is given by

$$E_{test} - E_{train} = k(h/P)^a$$

where $P$ is the number of training samples, $h$ is a measure of “effective capacity” or complexity of the machine, $a$ is a number between 0.5 and 1.0, and $k$ is a constant.

As the number of training images increases, this gap between the error rates decreases. Moreover, as the complexity $h$ increases, $E_{train}$ decreases. Thus, there is a trade-off between the decrease of $E_{train}$ and the increase of the gap. The optimal value of $h$ thus achieves the lowest generalization error $E_{test}$.

This is important to note for the purposes of choosing a learning algorithm in a neural network, but it is relevant here because it confirms the intuition that neural network success is improved by the addition of training data.

2.1 Collection

In order to collect a training and testing set of data for use in the development of the neural network, I adapted a Matlab function “imSelectRoi” developed by Andriy Nych [18] in order to allow for multiple regions within a manuscript image to be selected and stored as distinct jpeg images in a single directory. The function presents a GUI that allows for easy region of interest selection and saving. The images are then stored in a character class folder as specified by the user when calling the function.
In initial testing, the number of training images remained small, though it was steadily increased as character class increased. Both training and testing images were selected using this tool, all at once, and they were later distinguished as testing or training based on visual inspection. The manuscript images used to collect the data were processed images in the RGB colorspace. No compression or distortion of the image occurs when using the GUI for data collection.

The Latin alphabet employed in the Codex A features 21 characters. The text is written in uncial script, which is characterized by broad single stroke letters and the absence of word separation. The script does not have lower and upper case distinctions, which makes the character recognition task simpler by reducing distinct character classes.
Table 2.1: Number of Original Images in Training and Testing Sets for Character Classes

<table>
<thead>
<tr>
<th>Character</th>
<th>Training</th>
<th>Testing</th>
<th>Character</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>76</td>
<td>N</td>
<td>159</td>
<td>110</td>
</tr>
<tr>
<td>B</td>
<td>140</td>
<td>106</td>
<td>O</td>
<td>127</td>
<td>198</td>
</tr>
<tr>
<td>C</td>
<td>138</td>
<td>95</td>
<td>P</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>D</td>
<td>150</td>
<td>103</td>
<td>Q</td>
<td>64</td>
<td>68</td>
</tr>
<tr>
<td>E</td>
<td>105</td>
<td>53</td>
<td>R</td>
<td>150</td>
<td>79</td>
</tr>
<tr>
<td>F</td>
<td>59</td>
<td>54</td>
<td>S</td>
<td>100</td>
<td>114</td>
</tr>
<tr>
<td>G</td>
<td>72</td>
<td>42</td>
<td>T</td>
<td>96</td>
<td>148</td>
</tr>
<tr>
<td>H</td>
<td>59</td>
<td>42</td>
<td>U</td>
<td>87</td>
<td>145</td>
</tr>
<tr>
<td>I</td>
<td>147</td>
<td>134</td>
<td>X</td>
<td>24</td>
<td>39</td>
</tr>
<tr>
<td>L</td>
<td>95</td>
<td>86</td>
<td>Z</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>M</td>
<td>181</td>
<td>197</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.2 Artificial Training Data Creation

A problem specific to the task of performing handwriting recognition on the Codex A is the lack of training data available. While datasets are continually being compiled by researchers, no dataset of Latin uncial script currently exists in an open source form. Images of whole pages of medieval Latin manuscripts have been made available by the hosts of the ICFHR2016 Competition on the Classification of Medieval Handwritings in Latin Script, though they have not compiled a set of the individual characters [19].

Because training data collection is a time consuming task and cannot be automated without the use of an unsupervised learning system, it was desirable to produce artificial training data using the images that were hand selected. I outline the changes introduced in the hand selected data to compile this artificial data below.

2.2.1 Bilateral Filtering

In order to reduce background noise and introduce more defined character outlines, I employed a cartooning function developed by Douglas Lanman, available on Mathworks File Exchange, which utilizes bilateral filtering to abstract the input image.

Bilateral filtering replaces pixel values of an input image with weighted averages of the nearby pixel values. This weight is dependent not only on the Euclidean distance between pixels but also differences of intensity [20]. See Appendix A for a mathematical definition of the bilateral filter.
Using the bilateral filter, we may specify how many discrete pixel values, either in RGB or grayscale, we wish to appear in the output image. For the purposes of training data creation, I chose the number of quantization levels to be 4 in order to preserve the shape of the characters while also minimizing as much background noise as possible. I chose the half-size of the Gaussian bilateral filter window to be 5, and the standard deviation of filter for the spatial domain to be 3, and the intensity domain to be 0.1. In Figure 2.3, the bilaterally filtered training data images populate the middle row.

### 2.2.2 Principal Components Analysis

Principal components analysis, or PCA, at its simplest is a means of decreasing the dimensionality of a dataset while maintaining the important features of the data. PCA achieves this by first calculating the covariance matrix of the original set of observations, a matrix of pixel values in this case, and then finding the eigenvectors of this matrix. These eigenvectors are the principle components of the set of observations, and because these eigenvectors are orthogonal to one another, they can be used to re-base the coordinate system of the dataset. In other words, the x and y axes may be reoriented to the axes represented by these principle components [21]. For a more thorough explanation of covariance matrices and eigenvectors, see Appendix A.

PCA is especially helpful in the creation of artificial training data because it extracts the information that is the most relevant to classification. Moreover, the principal components from different images may be layered to produce more variation in the training set.

I adapted several Matlab programs developed and released by Matthew Dailey, licensed under a Creative Commons Attribution 3.0 License [22], which calculate the principal components of images and then layer these components to produce new images. In figure 2.3, you see on the final row a series of images created with this layering technique. Each image shown consists of the original image (top row) in gray scale projected onto the first order principal components of five other images in the training set. I iterated in increments of five over the first fifty principal components generated for each character class to produce the total number of images given in Figure 2.4. In the case of X and Z, only 20 and 5 principal components could be calculated, respectively, so image production was altered accordingly. I inspected the new training data visually, and if any image appeared malformed, I deleted this entry.
Figure 2.3: Training Data: Original, Quantized, and PCA Images

<table>
<thead>
<tr>
<th>Char</th>
<th>PCA</th>
<th>Bilateral</th>
<th>S,R,T</th>
<th>Char</th>
<th>PCA</th>
<th>Bilateral</th>
<th>S,R,T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>579</td>
<td>100</td>
<td>936</td>
<td>N</td>
<td>730</td>
<td>55</td>
<td>1582</td>
</tr>
<tr>
<td>B</td>
<td>644</td>
<td>75</td>
<td>1262</td>
<td>O</td>
<td>786</td>
<td>82</td>
<td>1156</td>
</tr>
<tr>
<td>C</td>
<td>571</td>
<td>104</td>
<td>1283</td>
<td>P</td>
<td>654</td>
<td>76</td>
<td>872</td>
</tr>
<tr>
<td>D</td>
<td>567</td>
<td>99</td>
<td>1350</td>
<td>Q</td>
<td>593</td>
<td>60</td>
<td>576</td>
</tr>
<tr>
<td>E</td>
<td>679</td>
<td>79</td>
<td>945</td>
<td>R</td>
<td>690</td>
<td>111</td>
<td>1373</td>
</tr>
<tr>
<td>F</td>
<td>641</td>
<td>45</td>
<td>532</td>
<td>S</td>
<td>676</td>
<td>66</td>
<td>900</td>
</tr>
<tr>
<td>G</td>
<td>628</td>
<td>52</td>
<td>603</td>
<td>T</td>
<td>612</td>
<td>67</td>
<td>857</td>
</tr>
<tr>
<td>H</td>
<td>641</td>
<td>47</td>
<td>531</td>
<td>U</td>
<td>345</td>
<td>40</td>
<td>811</td>
</tr>
<tr>
<td>I</td>
<td>879</td>
<td>119</td>
<td>1305</td>
<td>X</td>
<td>106</td>
<td>15</td>
<td>230</td>
</tr>
<tr>
<td>L</td>
<td>642</td>
<td>65</td>
<td>855</td>
<td>Z</td>
<td>7</td>
<td>4</td>
<td>56</td>
</tr>
<tr>
<td>M</td>
<td>839</td>
<td>89</td>
<td>1810</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Number of Training Images Produced in Artificial Training Production

2.2.3 Translations, Rotations, Skewing Data

In 2003, Simard, Steinkraus and Platt managed to achieve 99.7% accuracy rate for the MNIST dataset using a simple neural network with two convolutional-pooling layers, followed by a hidden fully-connected layer with 100 neurons, by increasing the training set, introducing translations and rotations and skewing the training data.

In order to introduce vertical and horizontal translations, I wrote a Matlab program that calculates the range for the average pixel values for the background of an image, and then randomly selects values from that range with which to pad the image. I introduced rotations of $\pm 5^\circ$, $\pm 10^\circ$, and $\pm 15^\circ$, using Matlab’s...
“imrotate” function, and I skewed the images using Matlab’s “maketform” and “imtransform” functions.

2.2.4 Manual Data Alterations

While the previous two methods help overcome the challenge presented by the sparse usage of some of the letters, there is the additional problem that some of the character classes appear in largely degraded form throughout the manuscript. In particular, “F” and “Z” were difficult classes for which to find pristine training data. For both of these classes, I manually altered some of the original training data that I collected, to make them appear visually more like the prototypes shown in figure 2.2. Some examples are shown in figure 2.4.

2.3 Pre-Processing

EBLearn offers the ability to pre-process training and testing sets to ensure standard sizes and channels. To reduce the dimensionality of the data, I converted both sets to grayscale, and I set the image size to be 32x32x1, with a Lp pooling method for resizing the images, with $p = 2$.

Lp pooling is a biologically inspired sub-sampling method that gives increased weight to stronger features and less weight to weaker ones. Pierre Sermanet, Soumith Chintala and Yann LeCun have demonstrated the superior performance of Lp pooling in neural net recognition tasks, in particular for the case where $p = 2$ and $p = 4$ [11]. Formalized, Lp pooling is given by

$$O = \left( \sum \sum I(i, j)^p G(i, j) \right)^{1/p}$$

(2.1)
where $G$ is a Gaussian kernel, $I$ is the input feature map and $O$ is the output feature map. It's worth noticing that for $p = 1$, this equation yields Gaussian averaging, and for $p = \infty$, it yields max pooling [11].
Chapter 3

Neural Network Architecture

3.1 Learning Method

Within the field of machine learning there are two major approaches to pattern recognition: supervised and unsupervised learning. In unsupervised learning approaches, the input is not labeled. Thus, the learning algorithm employed must discern the features of the data in order to group it into meaningful categories. Various methods exist to accomplish this including several types of clustering, neural networks, and Hidden Markov Models [23].

Because we are able to identify examples of each character class and definitively assert how many character classes exist within the Latin alphabet, a supervised learning method is more desirable for classifying characters in the Codex A. "Supervised" refers to the labeling of the input, such that errors in classification may be calculated and the system can learn from these mistakes [24]. We are employing a neural network architecture as the means of classification, so within the system, we have labeled each character class.

3.2 Net Model

The Lenet 5 neural network model, developed by LeCun, Bottou, Bengio, and Haffner in 1998, has been demonstrated to be effective at recognizing handwritten digits, achieving as low as a 0.95% test error rate for the MNIST dataset, and a 0.8% test error rate for the dataset by expanding the existing 60,000 image training set by 540,000 distorted images [3].

The neural net consists of six hidden layers that are fully connected. The first layer is a convolutional layer that consists of six feature maps. Each of these
feature maps performs a convolution operation on identical parts of the input image (32x32), using a 5x5 kernel of weights. After applying the kernel, a bias is added and a sigmoid function (tanh) is applied to produce the final feature map [3].

Following the first convolutional layer is a sub-sampling layer that reduces the 28x28 feature maps to 14x14 using a L2 pooling method. A sigmoid function is applied and a bias is also added post sub-sampling. Next comes a second convolutional layer with sixteen feature maps using a 6x6 kernel, then the second sub-sampling layer that reduces the 10x10 images to 5x5. The final layer is a fully connected layer which performs a linear combination of its input and their internal weights, adding a bias and applying a sigmoid. This layer’s results are put in the output, such that a vector containing a value, or energy, for each of the classes is produced. The error in the expected values for each of the classes is then calculated, and the backpropagation algorithm can be used to calculate the gradient of the loss function, with respect to each of the weights in the network. These gradients are used in the gradient decent optimization method to minimize the loss function, re-updating the weights within the system [3].

3.3 Network Parameters

Using the EBLearn C++ open source machine learning library, classification of the Codex A characters was undertaken. The Lenet5 model was employed with a learning rate of $\alpha = 0.0001$. The L2 sub-sampling method was utilized in the sub-sampling layers.
Chapter 4

Net Training & Results

4.1 Validation Sets

I tested each net using a validation set composed of data selected from the manuscript that were damaged or imperfect (these data underwent no pre-processing or alterations, beyond downsizing and a change to grayscale), noted henceforth as the “V1” testing set. In Figure 4.1, examples of images included in the V1 set are given.

I also tested the most successful nets with a second validation set composed of randomly selected images from the training set, noted as the “V2” testing set. The number of training and testing images for these trials are included in Tables 4.3 and 4.4. It should be noted that when employing the second validation set, the training set did not include those images chosen for the testing set.

4.2 Epochs

For each net that I tested, I saved the weights calculated for each epoch, or iteration, and I trained each net for a maximum of fifty epochs. I chose this max based on the observation that the training and validation success rates tended to level out at this number of iterations. The results from each test were saved to a text file, which I inspected for the highest testing and training success rates.

While there exists a strong correlation between the training and validation success rates, they are not perfectly correlated. This is demonstrated by figure 4.1. This graph shows the training and V1 success rates for the net “Cartoon,” whose training set consists of the original images collected from the manuscript and the bilaterally filtered versions of these images. While large dips in success
in the V1 set are generally accompanied by dips of a smaller scale in the training set, some V1 dips are not; for example, see epoch 34.
4.3 Overfitting & Underfitting

The imperfect correlation seen in the training and validation success rates is indicative of the different features represented in the training and testing sets. A concern for all neural network training is overfitting and underfitting, overfitting being training a net such that it only recognizes the minute features particular to a training set, and underfitting being training a net such that it does not recognize enough features of a training set. Both overfitting and underfitting lead to poor results in validation set recognition.

Conveniently, the highest success rates for the training set and V1 set for the Cartoon Net (see Figure 4.1) are both given by epoch 31 at 98.70% and 77.10%, respectively. The fact that the net could recognize images in the training and testing sets bodes well for the net—it’s unlikely that overfitting or underfitting is occurring here. However, if we consider a case where we might achieve 100.00% accuracy rates for our training set, and we saw a corresponding drop in the recognition rate, we might become concerned that overfitting was occurring. We might similarly be concerned if the net had extremely low recognition rates for the training and testing set, as this would likely be a case of underfitting.

4.4 Training Sets

I trained several nets using the architecture described in Chapter 3 with various training sets, in order to test the contributions of new data to recognition results. Figure 4.2 gives the best recognition rates generated by these nets, tested over fifty epochs. I tested each net using the V1 testing sets and the most successful net using the V2 set.

In “Full,” the original testing set was used without modification. “Z Full” was trained on the original set with the manually altered “Z” class, and similarly, “FZ Full” was trained on the original set with the manually altered “F” and “Z” classes. “Cartoon”’s training set included these altered images, the original set, as well as the bilaterally filtered images. “PCA”’s training set included the altered images, the original set, and the PCA images. “Shift”’s training set included the altered images, original set, and translated, rotated, and skewed images. “PCA+Cartoon” was trained on the altered images, the original set, the bilaterally filtered images, and the PCA images. “Cartoon+Shift” was trained on the altered images, original set, bilaterally filtered images, and translated, rotated, and skewed images. Finally, “PCA+Cartoon+Shift” was trained on the altered
images, original set, bilaterally filtered images, PC images, and translated, rotated, and skewed images.

### 4.5 Background Training

While the various nets I trained achieved classification success rates of the testing sets that might suggest a fairly accurate classification of new manuscript data, real-world performance introduces new variables for which we must account. Background noise is a feature of real-world data that can seriously affect classification of characters by a net, so it is desirable to train the net to recognize what is a character and what is not.

I trained the all nets shown in Table 4.1, with the exception of “Full” and “Z Full”, to recognize background images using a training set of 200 background images, and a validation set of 27 images. The V1 success rates for the nets are given in Figure 4.2. Theoretically, training the net to recognize background images should improve the practical success rates of the net when trained on manuscript images. However, due to the great variation present in background images, this class requires a great deal of training data in order to recognize new examples. I discuss possible means of increasing the training data for this class in the conclusion, though it is worth noting the effect that the inclusion of this class has on the success rates of the nets. Recognition rates are included for the background class are included in Table 4.3.

<table>
<thead>
<tr>
<th>Character</th>
<th>Training</th>
<th>Testing</th>
<th>Character</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1000</td>
<td>701</td>
<td>N</td>
<td>1000</td>
<td>1526</td>
</tr>
<tr>
<td>B</td>
<td>1000</td>
<td>1027</td>
<td>O</td>
<td>1000</td>
<td>1151</td>
</tr>
<tr>
<td>C</td>
<td>1000</td>
<td>1064</td>
<td>P</td>
<td>1000</td>
<td>644</td>
</tr>
<tr>
<td>D</td>
<td>1000</td>
<td>1166</td>
<td>Q</td>
<td>750</td>
<td>543</td>
</tr>
<tr>
<td>E</td>
<td>1000</td>
<td>805</td>
<td>R</td>
<td>1000</td>
<td>1316</td>
</tr>
<tr>
<td>F</td>
<td>750</td>
<td>511</td>
<td>S</td>
<td>1000</td>
<td>736</td>
</tr>
<tr>
<td>G</td>
<td>750</td>
<td>589</td>
<td>T</td>
<td>1000</td>
<td>558</td>
</tr>
<tr>
<td>H</td>
<td>750</td>
<td>521</td>
<td>U</td>
<td>750</td>
<td>506</td>
</tr>
<tr>
<td>I</td>
<td>1000</td>
<td>1363</td>
<td>X</td>
<td>200</td>
<td>175</td>
</tr>
<tr>
<td>L</td>
<td>1000</td>
<td>657</td>
<td>Z</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>M</td>
<td>1000</td>
<td>1919</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Number of Images in Training and V2 Testing Set for BG PCA+Cartoon+Shift
Chapter 4: Net Training

<table>
<thead>
<tr>
<th>Net Title</th>
<th>Epoch</th>
<th>Train (%)</th>
<th>V1 (%)</th>
<th>BG Train (%)</th>
<th>BG V1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>49</td>
<td>99.95</td>
<td>86.73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z Full</td>
<td>38</td>
<td>99.95</td>
<td>87.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FZ Full</td>
<td>45</td>
<td>99.95</td>
<td>88.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cartoon</td>
<td>25</td>
<td>99.97</td>
<td>87.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA</td>
<td>13</td>
<td>99.90</td>
<td>84.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>30</td>
<td>99.56</td>
<td>83.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA+Cartoon</td>
<td>19</td>
<td>98.35</td>
<td>82.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA+Cartoon+Shift</td>
<td>49</td>
<td>99.43</td>
<td>83.81</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Success Rates for Nets Selected for Highest V1 Recognition Rates

<table>
<thead>
<tr>
<th>Net Title</th>
<th>Epoch</th>
<th>Train (%)</th>
<th>V1 (%)</th>
<th>BG Train (%)</th>
<th>BG V1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG FZ Full</td>
<td>22</td>
<td>98.80</td>
<td>83.03</td>
<td>95.92</td>
<td>71.43</td>
</tr>
<tr>
<td>BG Cartoon</td>
<td>6</td>
<td>99.55</td>
<td>83.67</td>
<td>99.49</td>
<td>81.63</td>
</tr>
<tr>
<td>BG PCA</td>
<td>14</td>
<td>99.78</td>
<td>79.45</td>
<td>100.00</td>
<td>83.67</td>
</tr>
<tr>
<td>BG PCA+Cartoon</td>
<td>31</td>
<td>99.58</td>
<td>81.02</td>
<td>100.00</td>
<td>81.63</td>
</tr>
<tr>
<td>BG Shift</td>
<td>48</td>
<td>99.33</td>
<td>81.70</td>
<td>100.00</td>
<td>73.47</td>
</tr>
<tr>
<td>BG Cartoon+Shift</td>
<td>28</td>
<td>99.60</td>
<td>81.02</td>
<td>100.00</td>
<td>79.59</td>
</tr>
<tr>
<td>BG PCA+Cartoon+Shift</td>
<td>30</td>
<td>99.18</td>
<td>83.72</td>
<td>100.00</td>
<td>75.51</td>
</tr>
</tbody>
</table>

Table 4.3: Success Rates for Nets Trained with Background Images Selected for Highest V1 Recognition Rates

4.6 Results

After training three nets for each testing set, I chose the epoch results which yielded the highest V1 success rate amongst the three nets, given in Tables 4.2 and 4.3. The training set that produced the most accurate net by the V1 success rates is "FZ Full," achieving a test success rate of 88.19%. Somewhat surprisingly, "PCA," "PCA+Cartoon," "Shift," and "PCA+Cartoon+Shift" all performed at least three percentage points lower for the V1 set, and “Cartoon” performed less than a percentage point lower. However, once nets were trained using background images, "BG PCA+Cartoon+Shift" took the lead in V1 success rates, achieving 83.72% success rate for the V1 set. I tested this net using a V2 testing set, and achieved a 99.49% success rate for the training set and a 98.27% success rate for the V2 set on the 24th epoch.

Recognition rates for the background class are included in Table 4.3 as well. It appears that on average, the V1 recognition rate is lower than the mean V1 rate for all the character classes, with the exception of the background success rates for “BG PCA” and “BG PCA+Cartoon.”
4.7 Manuscript Testing

These results give us some intuitions about what the relative success of the nets application to real world data might be; however, the testing sets only approximate the variety we actually find in the manuscript itself. I undertook the task of detecting the identity of new images taken from the manuscript, using EBLearns “detect function. EBLearn allows users to pre-process input images using the same pre-processing parameters specified for the training data and lets one specify the maximum, minimum, and number of scales desirable for the net to attempt to classify. I attempted to classify representative and well-formed single characters in sub-regions of the manuscript images, specifying the maximum scale to be 0.5 and the minimum scale to be 0.3. The step factor between the minimum and maximum I specified to be 1.2.

I chose well-formed single characters to classify because they represented the simplest classification problem and they required less computing power than images including multiple characters. I ran the detect function on images from each character class that were of relatively small size, typically between 64x64x3
pixels and 45x45x3 pixels. I found that the net recognized larger regions of the input images when I limited the size of the scales for this particular input size, though I was not able to achieve any successful classifications for these new input images. In response to these poor results, I varied the scaling parameters, changing the maximum scale to 2, 1, and 0.25, the minimum scale to 1, 0.5, and 0.25 and testing every combination of these two parameters. However, the net misidentified smaller portions of the images when these alternate parameter values were used. I discuss the practical limitations of the net in Chapter 5 and the potential causes of its failure to recognize new examples. Somewhat encouragingly, the net was able to detect that characters did exist in many of the images, as summarized in Table 4.4. I included the character identity that had the highest energy associated with its classification. “H”, “M”, and “P” were the most popular identities with individual recognition rates of 83.33%, 76.77%, and 86.08%, respectively. “P”’s recognition rate is the only one greater than the mean recognition rate, so it appears that the net’s classifying new data is not simply a function of those characters it best identifies.

<table>
<thead>
<tr>
<th>Char</th>
<th>Recognized (Y/N)</th>
<th>Classification</th>
<th>Char</th>
<th>Recognized (Y/N)</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>Y</td>
<td>M</td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>H</td>
<td>O</td>
<td>Y</td>
<td>H</td>
</tr>
<tr>
<td>C</td>
<td>N</td>
<td>P</td>
<td>Y</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Y</td>
<td>H</td>
<td>Q</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>E</td>
<td>Y</td>
<td>I</td>
<td>R</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>N</td>
<td>S</td>
<td>Y</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>N</td>
<td>T</td>
<td>Y</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Y</td>
<td>H</td>
<td>U</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Y</td>
<td>H</td>
<td>X</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>N</td>
<td>Z</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Success of Character Identification in New Manuscript Image Data
Chapter 5

Conclusion

5.1 Findings Regarding Training Sets

Interestingly, while the neural network trained with the PCA+Cartoon+Shift training set yielded the highest recognition rate for the set of nets trained to recognize background data, the same net yielded a lower recognition rate among the nets trained without background data. In fact, the set of nets not including background data in the training data does not precisely predict the ordering of the most successful nets among the set trained on background data, as one might expect it to. It appears that the additional parameters introduced by the background class causes fluctuations in the success and relative success of the training set classes. The recognition rates for the background class, while sometimes as much as twelve percentage points lower than the mean V1 success rate for all the character classes of a given net, were not so low that they could entirely account for the lower mean V1 classification rate, however. The results instead seem to point to the interconnected nature of the neurons and the difficulty in predicting the effects they will have on one another, due to the sheer number of parameters involved. Moreover, the differences in relative success of the nets based on the inclusion of the background class suggest that a wider variety of training data is useful in discerning between character data and background, an unavoidable task inherent to the classification of real-world data.

As far as deducing what artificial training data is most useful in improving the success rates of the nets, it appears that the manual alterations yielded the most improvement, with a 1.73 percentage point increase for the V1 set among the group of nets not trained on background data. In combination, bilateral filtering, principle components analysis, and translations, rotations, and skewing produced
a training set that yielded modest improvement over the use of bilateral filtering alone; however, each of these artificial sets (with the exception of the bilaterally filtering) individually acted to decrease the recognition rate established by the “BG FZ Full” net.

It’s important to note that the V1 testing set itself is an approximation of the variety of character data a net might encounter. The meaningfulness of the relative success of each of these nets is thus dependent on the quality and variety of the data captured by the V1 set. While I sought to include examples that were both degraded and easily discernible, the fact that all the images included in this set were not training data quality might mean that certain artificial training data images would more useful in picking out the features represented. I operated under the assumption that a net that included the most training data that also yielded the highest recognition rate would produce the best results for real-world data.

5.2 Possible Means of Improvement

While relatively successful results were achieved for the V1 testing sets, there are means by which this research could be improved upon, especially with regard to resources in the form of time and hardware. I also briefly discuss potential means of improving manuscript recognition results in the section below entitled, "Manuscript Testing Limitations."

5.2.1 Computing Power

The training and application of neural networks require a fair amount of computing power. I was able to train networks with 20000 plus parameters and over 12500 training images on a computer with 8 GB of RAM, though I was unable to complete detections of background regions or characters in images of the manuscript. While a supercomputer would have been ideal, due to a software package incompatibility with the University of Mississippi Supercomputing Center’s operating system, that was not an option for this research. I secured a 32 GB computer from the UM Computer Science Department, and while I was not able to complete detection of characters for images of entire manuscript pages, I was able to run detections on small regions of interest with a limited number of scaling sizes. Obviously, running detections on entire manuscript pages would be preferable, especially for applications requiring diplomatic texts. For these projects, more computing power would be required.
5.2.2 Training Data

If I had the time and resources to create a larger training data set from manuscript image data, net success rates would likely have risen. Datasets such as the MNIST dataset includes 60,000 training images and 10,000 testing images for ten classes. The training sets I used had a fraction of that data, for over twice as many classes. While I tried to compensate for the lack of data with artificial training data, the increase in recognition results for the V1 set was unfortunately not indicative of success for implementation on new manuscript data.

Moreover, because computer power limitations prevented running detections on large background images, I was unable to expand the background training set by running the net on entire manuscript page image and adding misclassified background in the training set. More background training data, collected either manually or by bootstrapping methods would have been beneficial in reducing the instance of false-positives and improving recognition results for all character classes.

5.3 Manuscript Testing Limitations

While the testing sets revealed improvement in recognition results with the addition of artificial training data and modification of the original data (in the case of F and Z character classes), the net is not robust enough for use on new manuscript data. This is most likely attributable to the limited amount of training data used, though it is also possible that results could be improved by averaging the weights of multiple trained neural networks, or by employing a different neural architecture. Further background image training would also improve results, as discussed in the previous section.

5.4 Applications

While practical application of this net to manuscript data is not yet feasible, several projects and potential applications of this research provide an impetus to extend its result. In particular, the transcription and manuscript imaging processes in many cases would benefit from an automated means of text recognition. Beyond the applications of the net, compiling the training and testing sets for this project is itself an important step in making automated handwriting recognition in manuscripts a more feasible endeavor. Collection is time consuming,
and diversity of training data is desirable for creating a net that could be used by many different scholars for many different manuscripts. Currently, there is no manuscript training data repository available, and while handwriting recognition software exists for many language of the world, the distinctive hands and styles present in pre-Medieval and Medieval manuscripts are often not accounted for in these approaches.

In the hopes that other scholars may use the training and testing data I’ve collected in their own research, all the data used in this project will be made publicly available through the Lazarus Project website.

5.4.1 Transcriptions

While this research only represents the first step in creating an end-to-end handwriting recognition system, there are important applications within the field of manuscript text recovery that warrant further research. As I mentioned in the introduction, damaged and malformed allographs are of particular interest, especially in the context of transcribing manuscripts. Because the Codex A consists of the Latin Gospels, we can consult with the older Greek texts as well as the other Latin Gospels to confirm text identities, so visual allograph identification is not an impossible task. However, in many manuscripts, such identification is inconclusive, so the motivation remains. Furthermore, an automated handwriting recognition system would be useful for conserving time and resources of visual transcribers and would also serve as a way to confirm visual identifications should this visual method still be desirable to employ.

5.4.2 Image Processing

For certain manuscript imaging projects, it is desirable to have a means of performing handwriting recognition as a part of the image processing work flow. Scholars working in St. Catherine’s Monastery, located in the Sinai Desert and home to a collection of manuscripts from the first millennium CE [25], have expressed their desire to have an automated means of transcription in order to decide the best means of imaging a particular manuscript. By analyzing the amount of text recognizable in various processed images of a single page that is representative of the damage seen in the rest of a manuscript, they could determine more quickly the appropriate parameters for the imaging and processing of the whole manuscript, saving time and resources.
Chapter 5: Conclusion

The manuscripts at St. Catherine’s Monastery are written in ten languages–Greek, Syriac, Georgian, Arabic, Christian Palestinian Aramaic, Latin, Caucasian Albanian, Armenian, Slavonic, and Ethiopic [25]. Each language would require its own neural network, though with the extensive training data available at the monastery, in a variety of hands and styles, more powerful neural networks could be developed, such that they would not have only manuscript-specific recognition abilities.
Appendix A

Mathematical Foundations

A.1 Bilateral Filter

The bilateral filter is defined by

\[ I_{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \]  

(A.1)

where the normalization term

\[ W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \]  

(A.2)

ensures that the filter preserves image energy and

\( I_{\text{filtered}} \) is the filtered image;

\( I \) is the original input image to be filtered;

\( x \) are the coordinates of the current pixel to be filtered;

\( \Omega \) is the window centered in \( x \);

\( f_r \) is the range kernel for smoothing differences in intensities.

\( g_s \) is the spatial kernel for smoothing differences in coordinates.

Consider a pixel located at \((i, j)\) that has a neighbouring pixel located at \((k, l)\).

The weight assigned for pixel \((k, l)\) to denoise the pixel \((i, j)\) is given by:

\[ w(i, j, k, l) = e^{\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma^2_d} - \frac{\|I(i,j) - I(k,l)\|^2}{2\sigma^2_r}\right)} \]  

(A.3)

where \( d \) and \( r \) are smoothing parameters and \( I(i, j) \) and \( I(k, l) \) are the intensity of pixels \((i, j)\) and \((k, l)\) respectively. After calculating the weights, these intensities are normalized. The denoised intensity of pixel \((i, j)\) is given by
\[ I_D(i,j) = \frac{\sum_{k,l} I(k,l) \ast w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)} \]  \hspace{1cm} (A.4)

A.2 Covariance Matrix

Given \( n \) sets of variates denoted \( X_1, \ldots, X_n \) and \( \mu_i \) as the mean, the first-order covariance matrix is defined by

\[ V_{ij} = \text{cov}(x_i, x_j) = < (x_i - \mu_i)(x_j - \mu_j) > \]  \hspace{1cm} (A.5)

A.3 Eigenvector

Define an eigenvector as a column vector \( X_R \) satisfying

\[ AX_R = \lambda_R X_R, \]  \hspace{1cm} (A.6)

where \( A \) is a matrix, so

\[ (A - \lambda_R I)X_R = 0, \]  \hspace{1cm} (A.7)

which means the eigenvalues must have zero determinant, i.e.,

\[ \text{det}(A - \lambda_R I) = 0. \]  \hspace{1cm} (A.8)
Bibliography


