Trinity College Dublin

The University of Dublin

School of Computer Science and Statistics

Investigating the applicability of neural networks in the recognition of handwritten traditional Irish script.

Patrick Meleady

Research Supervisor: David Gregg

A dissertation submitted in fulfilment of the requirements for the degree of Master in Computer Science

Submitted to the University of Dublin, Trinity College, April, 2019

Declaration

I, Patrick Meleady, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signed: _____ Date: _____

Acknowledgements

I would like to thank my research supervisor for all the guidance and help that he provided throughout the course of this dissertation, without him there would not be a dissertation to submit.
I would also like to thank the School of Computer Science and Statistics for the last five years and for providing me with the knowledge necessary to pursue a career in Computer Science.

Abstract

Language preservation is the effort to prevent languages from becoming unknown. Language preservation is widely practiced as the death of a language can result in the loss of the speaking population's culture, oral traditions and other inherited beliefs. In order to preserve a language, an effort can be made to preserve the language's manuscripts through transcription.
As the number of manuscripts can grow to large numbers, manual transcription can be both expensive and time-consuming. As such, an automatic process is required. In the case of traditional Irish script however, no such research has been conducted in creating this automated process.
Neural networks have a wide range of applicable uses, several of which involve pattern recognition. Due to the suitability of neural networks in pattern recognition, they have seen an increased use in text transcribers and handwriting recognisers. This work aims to show applicability of neural networks in creating an automatic transcription tool for handwritten traditional Irish script.
The report details each step in the created pipeline, justifying its presence while also mentioning various researched methods which were not suitable for this application. The work then makes use of the created pipeline in order to train a neural network and evaluate its performance in order to determine the applicability of the solution.
By analysing the performance of the trained neural network, this work shows that applications exist for neural networks in the field of handwritten traditional Irish script recognition. The work also provides conclusions based on gathered results, on how to improve the performance of the network moving forward.

Contents

List of Figures

1 Introduction

In this chapter, the main motivations behind this research are outlined. This chapter also defines the research question along with the set of aims and goals which determine the scope of the research. Finally this chapter details the structure of the dissertation chapter by chapter, detailing the main points covered in each section.

1.1 Motivation

As mentioned in the previous chapter, the decline of a language can cause the loss of some of the defining characteristics of the speaking population. As such, efforts are made in order to prevent this loss occurring. The decline of a language can be stemmed or prevented through several different means, such as the encouragement of younger generations to speak the language or the preservation and publication of the language's documents and manuscripts. The preservation of manuscripts faces several issues which limit their availability to the speaking population due their increasing frailty and declining structural integrity over time. A solution to this issue is to digitally transcribe these manuscripts in order to preserve the manuscript itself while providing the digital transcription to the general public. The question then arises of how to digitally transcribe the manuscripts in a cost effective manner. Manual transcription of these documents can be extremely costly, as the act requires transcribers who are fluent in the language in order to effectively and accurately transcribe the language. These transcribers can be exceptionally rare as the effort to preserve a language naturally implies a small population of speakers. This cost then scales with the

amount of manuscripts. Volunteers can be used in order to increase the speed at which documents are transcribed however this introduces quality concerns over the transcription and as such, requires quality control mechanisms to be put in place. Overall, manual transcription can prove to be not only costly in execution but also not preserve a high level of quality when the repository of documents is large.

A solution to this problem is the creation of an automated transcription process. The automated process would involve locating and recognising the handwritten text on the manuscript and outputting the transcribed text to a text file.

As of the time of writing, no research has been conducted in the creation of this automated process for manuscripts containing traditional Irish script. The creation of this process however would benefit several organisations who curate and collect examples of handwritten traditional Irish script. At present these organisations host a volunteer based approach for the manual transcription of the manuscripts which are susceptible to the weaknesses discussed previously.

The creation of an automatic process for the transcription of handwritten traditional Irish script would improve the overall quality and speed of transcription for approximately 500,000 manuscript pages across these repositories.

1.2 Research Question

This dissertation aims to explore the applicability of neural networks in the recognition of handwritten traditional Irish script.

1.3 Research Aims

The following aims can be derived from the research question in order to define the scope and goals of the dissertation:

1. Explore the feasibility of a neural network based solution for the recognition of

handwritten traditional Irish script.

2. Define a pipeline for the training and evaluation of the neural network based approach.
3. Build a training set and use the defined pipeline to create a neural network model.
4. Evaluate the created model in terms of the size of training set in order to estimate future work.

1.4 Dissertation Structure

This dissertation is divided into five different chapters, the main contents of the following chapters are as follows:

Chapter 2 provides a background into the major sources used for the dissertation along with examples of related work. The history of traditional Irish script along with the background of the main collection from which samples were retrieved is described here. Tesseract Kay 2007, which is the main tool used for training neural networks is also described in this chapter. This chapter also then refers to related works and how these works were incorporated in this dissertation.

Chapter 3 details the implementation created for the dissertation. This chapter outlines the design of the created pipelines and each stage within them. The chapter details the preprocessing pipeline, the training pipeline and the evaluation pipeline while going into further details on the methods within them. Different concepts such as binarisation and noise removal along with levenshtein distance are explained fully in this chapter along with researched topics which were not suitable for this proposed pipeline.

Chapter 4 details the evaluation of the trained network. This chapter explains the overall evaluation methods and presents the performance of the trained network against different criteria. The results are then analysed in order to point out possible shortfalls which could be circumvented in future work in order to improve the performance of the trained network.

Lastly, Chapter 5 details the overall findings of the dissertation along with reference to the

aims presented in section 1.3. This chapter then presents the answer to the research question with reference to the generated results and suggestions for future work in this area.

2 Background

In order to correctly assess the feasibility of neural networks in the recognition of handwritten traditional Irish script, it is important to not only be familiar with any previous research in this field but also in the field of handwriting recognition as a whole. The processes and approaches taken by other researchers will provide valuable insights into the overall pipeline and methodology. Accompanying the insights provided by other researchers, it is also important to examine traditional Irish script in order to identify not only the key differences between it and its modern counterpart. This examination can also bring to light certain difficulties which may arise throughout the creation of the handwriting recogniser. These difficulties can then be taken in to consideration during the design of the pipeline. Overall this prior research can aid in not only the understanding the core problem, but also can provide a starting foothold in creating a pipeline by adapting how other researchers have tackled this problem for other languages and scripts.

2.1 Traditional Irish Script

2.1.1 Overview

Prior to the 1940's, most written Irish was written using traditional Irish script. This script makes use of the Gaelic type and differs from the written form that is in use today. Traditional Irish script makes use of an eighteen letter alphabet which is composed of letters from the Latin alphabet excluding English letters such as j,v,w,y and z and Latin letters such

as x and q. In addition to these eighteen characters, each vowel can also be represented in *fada* form. One major difference between the traditional and modern variants is the use of lenited consonants in the traditional form, these consonants are represented through the use of a dot above the consonant in question and are instead replaced in the modern form by the insertion of a 'h' character after the consonant. Nine lenited consonants exist in the script which bring the total amount of lower case characters to thirty two.

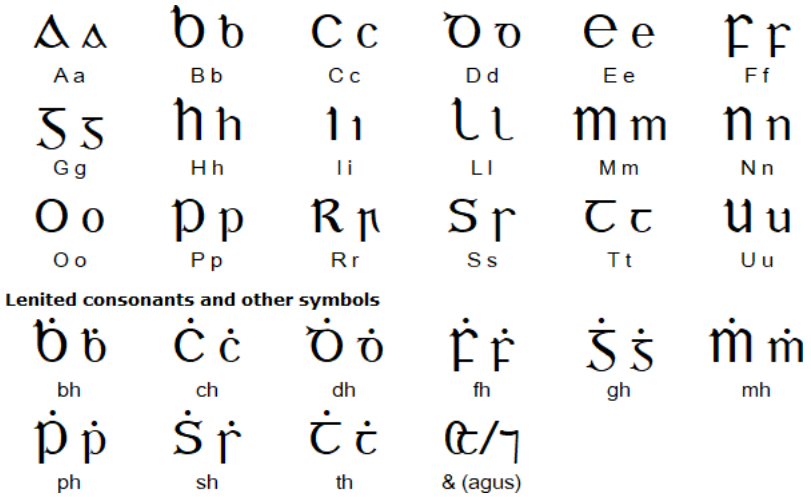


Figure 2.1: The traditional Irish alphabet in Gaelic type (Taken from: https://www.omniglot.com/images/writing/clogaelach.gif)

2.1.2 Potential Difficulties

Similar characters The existence of these lenited consonants may provide a challenge to the recogniser as these consonants are near identical to their respective non-lenited form. This may lead to the misclassification of these characters with their non-lenited forms or cause the reverse where a non-lenited character is misclassified due to noise present above the character. In order to differentiate these separate characters, an effort will have be made to include several examples of the lenited and non-lenited forms in order for the neural network to become familiar with both forms. This is also true of the fada forms as they are also similar to their vowel counterparts and could be misclassified due to noise being misconstrued as a fada.

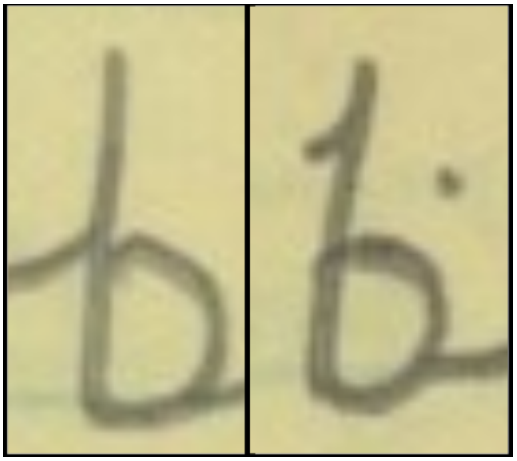


Figure 2.2: A comparison between a non-lenited and lenited b character

2.2 Neural Networks

A neural network is a computational model that attempts to approximate a function f(x) = y based on input and output examples (Sexton 2017). This model primarily focuses on solving two types of tasks, those being *regression* or *classification*. If the output set is a discrete set of classes, then the problem is a classification problem. One example of a classification problem is the recognition of specific characters in an alphabet (**class**). Regression then aims to output a continuous value based off the input. A classic example is predicting house prices based on location, size and other features (Sexton 2017). As the underlying problem that this project is trying to solve is a classification problem, the following description of neural networks will focus on implementation which also solve classification problems. It is important to note however that the underlying structure is identical for both problems and it is only how the output is interpreted that differs.
At its most basic, a neural network is a classifier with an input and output layer. The input layer takes in data from a given source and the output layer outputs a specific classification. Nodes in the network can be connected and pass their values on to the next layer. Incoming data from previous layers can be transformed via weighting functions. Additional transformations upon the data can then be achieved by adding more layers to the network. These hidden layers are used in order to transform data into an easily classifiable state. In figure 2.3, the general layout of a neural network can be seen, showing how information is

passed from the input layer, through the various hidden layers to the output layer.

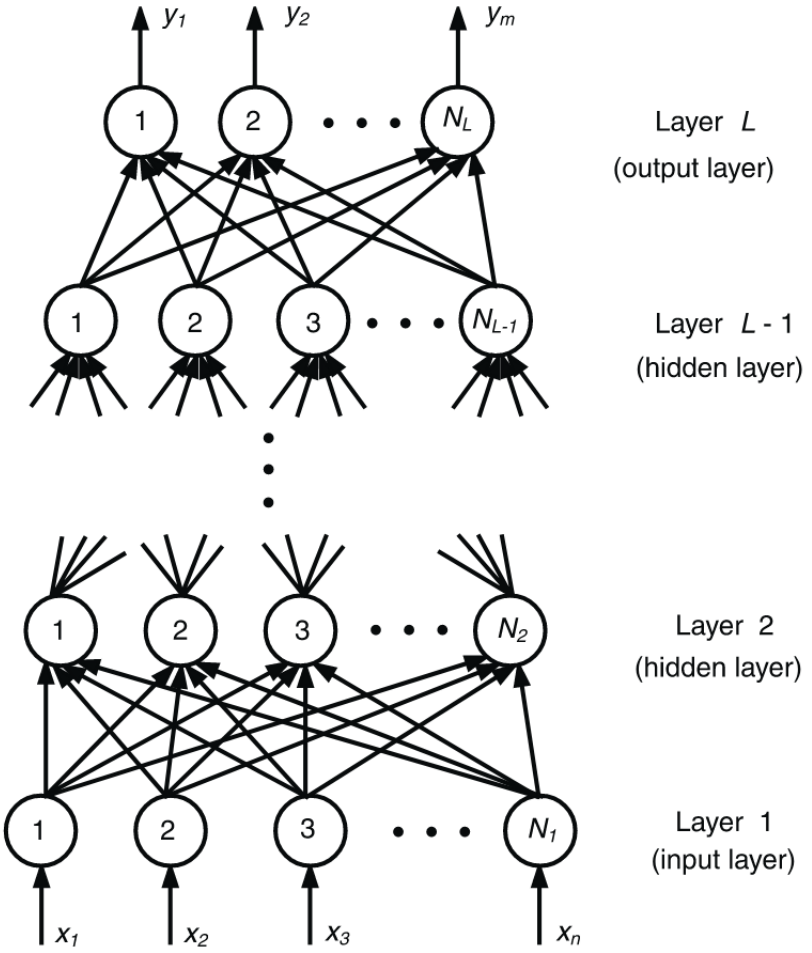


Figure 2.3: Layout of a neural network showing input and output layers, with nodes taking input of previous layers and passing them onto output. (Taken from: https://www.ieee.cz/knihovna/Zhang/Zhang100-ch03.pdf)

2.2.1 Recurrent Neural Networks and Long Short Term Memory Networks

While traditional neural networks have proven to have an extremely broad range of applications (Zhang 2000), limitations are present in their design. One limitation is that given the fixed structure of the network, valid input must be of a specific length. This requires that any input data not of the required length be either extended or compressed to the required length in order to considered as valid input. This limitation is also present if the input data is sequential. Sequential data must first be transformed into a single vector in

order to be processed. This transformation may cause a loss in data and also removes the concept of time if the input is considered as one single object.
It is with this limitation in mind that recurrent neural networks (RNNs) were originally created. A feedback loop is added to the model which improves the networks ability to handle sequential data. With the feedback loop, the network is able to better handle sequences of input by applying what has come before in a sequence to the current set of input. This feedback loop is similar in implementation to a series of connected neural networks, each taking the result of the previous network into account when processing its entry in the sequence. Overall this feedback loop improves the RNNs ability to solve problems (H.T. Siegelmann 1995). One problem with RNNs however is that long term dependencies are lost as the network progresses through the sequence. Dependencies which are recognised early in the sequence are less likely to affect later entries.
In order to tackle the issue of dependencies being lost over time, long short term memory (LSTM) networks were created (Sepp Hochreiter 1997). LSTM networks are similar to RNNs where a feedback loop is present which allows previous input to be reentered into the network. The main difference being that the LSTM networks allow for the reentered information to be modified before being passed into the network. The amount of modification done to the information is also differentiable allowing for the selective propogation of dependencies over the entire sequence of data. This network structure improves not only the traditional neural network's ability to handle sequenceable data but also the RNN's ability to handle dependencies over long sequences.

2.2.2 Tesseract

Tesseract is a command line tool used for text transcription. Tesseract was originally developed as proprietary software by Hewlett-Packard in 1994 however did not see major development apart from porting the code from C to C++ in 1998. In 2005 Hewlett-Packard released the software to the general public and marked the software as open-source. In 2006, Google sponsored the future development of Tesseract and continues to sponsor it to this

day. Originally Tesseract only featured support for English, however future versions have expanded this support to over 116 languages as of version 4.
On October 29th 2018, Tesseract released its first stable build of version 4, which introduced a new LSTM based OCR engine which employs the use of neural networks in its implementation. This implementation also allowed for other developers to download the library and train this network for new languages or scripts with a variety of different tools. The option was also open to developers on whether a new language should be trained on top of an already existing and supported language in order to reduce the amount of training time and the size of the training set required. This feature allowed developers to extend already trained models or in the case of handwriting recognition, add a new script to an already existing language.

2.3 School's Collection

The School's Collection is a collection of approximately 740,000 pages of folklore and local tradition and was compiled by pupils from 5,000 different across Ireland between 1937 and 1939. This collection provides a massive collection of handwriting in traditional Irish script. The collection itself is currently maintained by the national folklore collection and is fully available online. Due to its size, the collection will serve as the primary source of images as it contains not only images from over 5000 different sources, but also up to a total of 300,000 possible source images. This size of dataset ensures that a wide range of images are present in the training set which can improve the overall performance and ability to deal with varying input. Another key factor of the collection is that approximitely 45% of the images have been manually transcribed. The images are then presented in the collection along with a transcription if one exists. This transcription will serve as the groundtruth or the ideal output of the overall pipeline and is just as necessary as the source image itself. This combination of source image along with accompanying transcription provides nearly all of the required input in order to begin the training process.

3 Implementation

This chapter outlines the design and implementation of the training pipeline, detailing each step of the process and its role in the overall implementation. Throughout this chapter, the tools and resources outlined in the previous chapter will be used in order to train a neural network which will be ready for evaluation.

3.1 Dataset Collection

3.1.1 Dataset Characteristics

In order to effectively train a neural network, a dataset must first be constructed with which to train the network with. This dataset must be extensive and contain as many use cases as possible to allow for the neural network to familiarise itself with the maximum amount of possible scenarios. This dataset must also accurately represent the data that the neural network will come across once training is finished. These factors will determine the overall effectiveness of the final result, therefore it would be best to gather data from areas where the recogniser could potentially be used if the research is successful.
As the aim of this research is to train an OCR to recognise handwriting, which varies from writer to writer, the need for a dataset which provides several different examples of each possible character is extremely important in order to avoid overfitting for a particular handwriting style. A dataset must contain both a valid input and output with which to train the network. The input must take the form of images which contain handwritten traditional

Irish script and the output must be the transcribed text in a machine readable format.

3.1.2 Site Scraping and Separation

The first step in creating an effective dataset will be to transform the collection into a machine readable dataset fit for use. This will require parsing of each page in the School's Collection in order to determine its transcription percentage along with its language. This process will involve iterating over each page of the collection in order to first determine the language of the written script as well as examining the transcription amount on each entry. Each page which satisfies both conditions will then be scraped along with its corresponding transcription in order to be used in the dataset.
The transcription percentage is listed alongside with each page allowing for easy discrimination between fully and non-fully transcribed pages. Once the transcription of a page has been found, the transcription can then be inputted into a script which recognises the language of the inputted text. If the language outputted is Irish, the page can then be scraped and downloaded.
Once all pages from the collection have been parsed and the fully transcribed entries have been collected along with the respective transcriptions, a corpus can then be created pairing each image with a corresponding piece of text. This can then be used as input to the OCR and a groundtruth with which to judge the output against

3.2 Preprocessing

Once all scraping has been completed and a sufficiently large corpus has been created, the next step in creating a dataset will be to perform preprocessing operations on the dataset in order to simplify the input data. This is done to facilitate faster learning and to provide some uniformity to the otherwise extremely variant input. This section will explain the preprocessing options which are available and the illustrate how the final preprocessing pipeline was created.

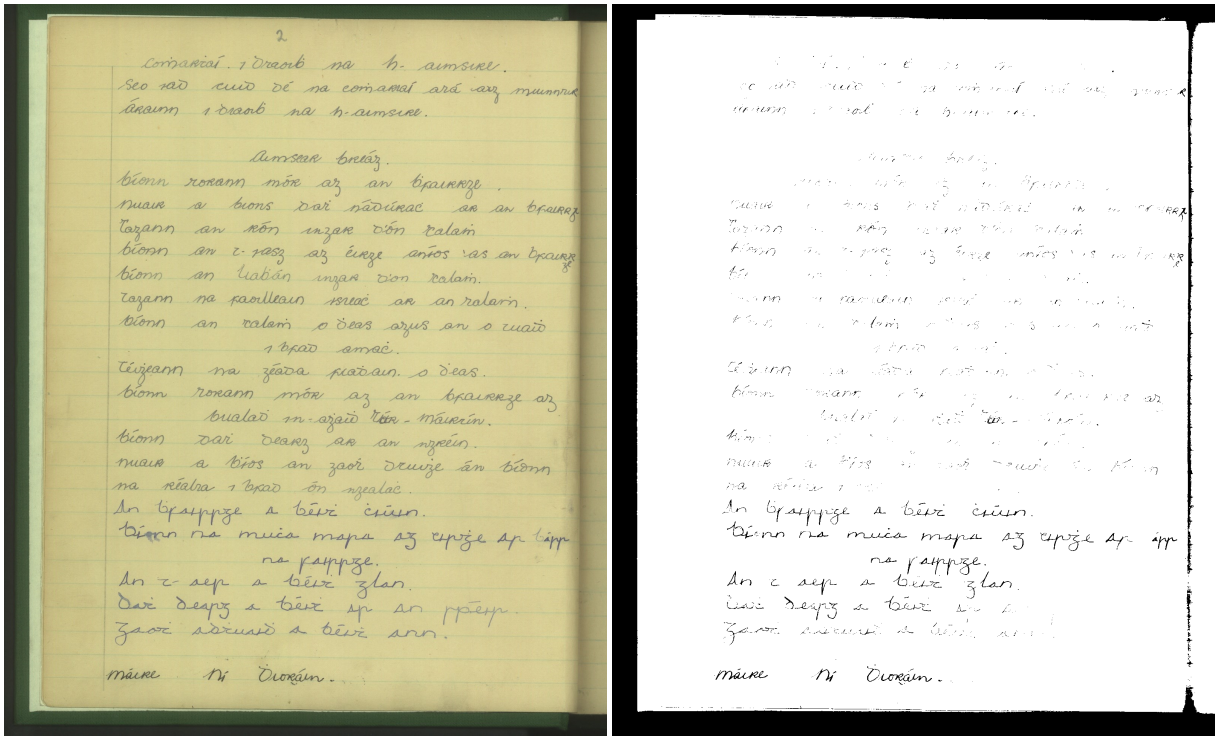
3.2.1 Binarization

One extremely efficient method to reduce the amount of data required to process is to binarize the input image. This process converts the image into a binary image containing only black and white pixels. This is extremely helpful as it clearly defines and separates foreground and background pixels while also clearly defining all edges. This can then lead to easier feature detection in specific characters.
In order to binarize an image, a threshold must be defined. Individual pixels can then be compared to the threshold in order to determine whether a black or white pixel should be used in their place. The thresholding of document images is still an unsolved problem due to different types of document degradations, such as uneven illumination, image contrast variation, bleeding-through, and smear(Tan 2011). In order to combat these difficulties, several methods for determining an appropriate threshold for a given image have been created which are applicable in different circumstances.

Global Thresholding

Global thresholding is by far the simplest method of converting an image into a binary image. This method picks a single threshold which is then applied to the entire image. This method assumes that the image in question has a bi-modal distribution and that there exists some threshold that will accurately separate the image. There exist several methods of determining this threshold.

Mean Threshold Mean thresholding is one of the simplest methods to select a threshold, this process iterates over each pixel and calculates the mean brightness. This mean is then used as the threshold. While this method might be simple, it is also susceptible to noise and other factors which may skew the mean and cause a loss of data. As shown below, this method does not perform well over the entire image due to varying light levels in different areas of the image.

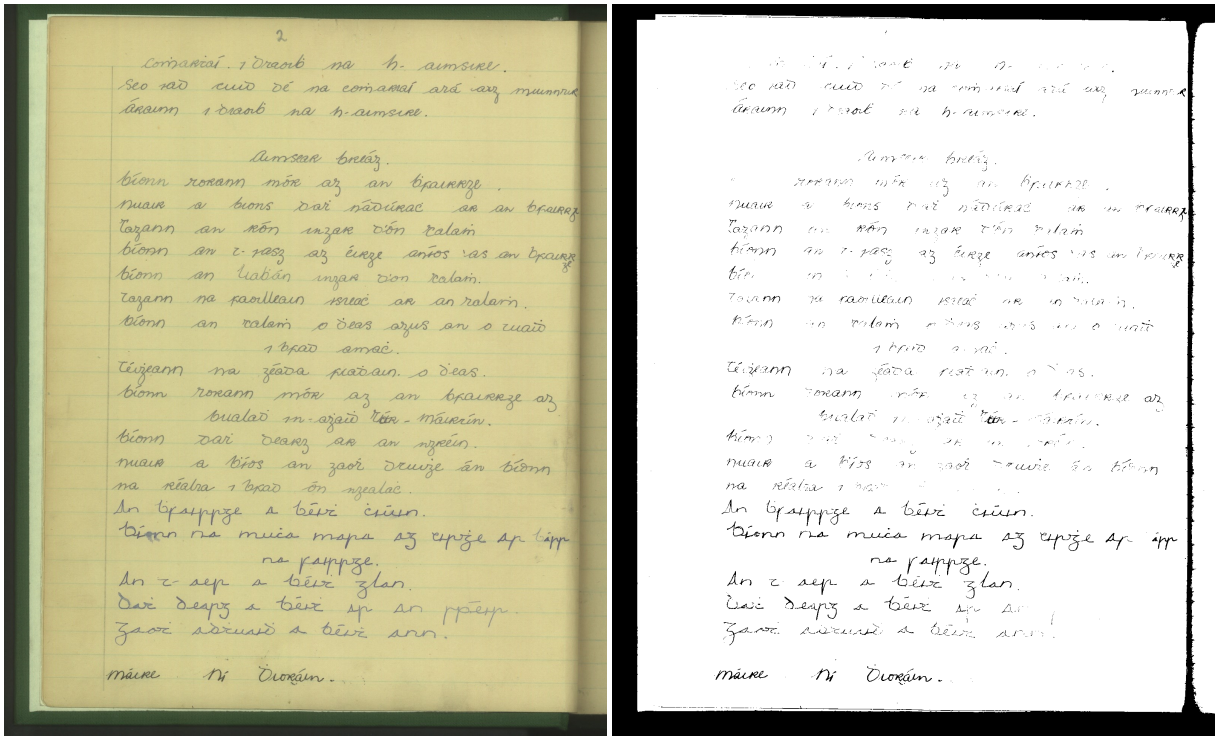


(a) Sample Page (b) Result of Mean Thresholding

The method

Otsu's Threshold Otsu's threshold is one method of determining a threshold for global thresholding. Otsu's method is to learn a threshold that can maximize the between-class variance or equivalently make light of the within-class variance of the entire image(Khan 2018). In order for Otsu's method to be successful, all assumptions for determining a single threshold are made (i.e. that the image has a bi-modal distribution, and that there exists enough space between the two distributions in order to exhibit bi-modal traits). Otsu's method determines a threshold by exhaustively iterating over each possible threshold while calculating the inter-class variance with the following algorithm. The algorithm then selects the threshold value which maximises the inter-class variance (which by extension minimises the intra-class variance). Otsu's method is limited however if the valley between the two distributions is not steep enough and as such can be affected by high levels of noise which may skew the mean variance.

Similar to binary thresholding, Otsu thresholding provides a threshold which is not suitable. While the method segments the page nicely, certain passages of text are starting to fade due

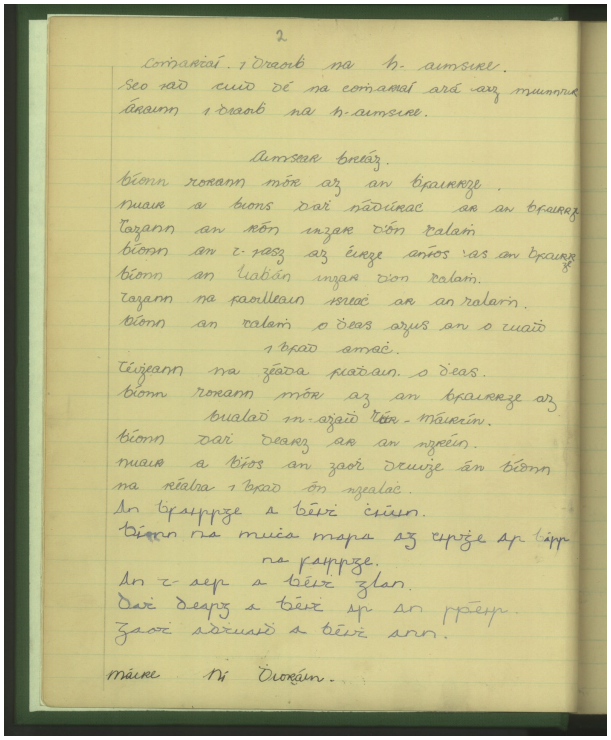


(a) Sample Page (b) Result of Otsu Thresholding

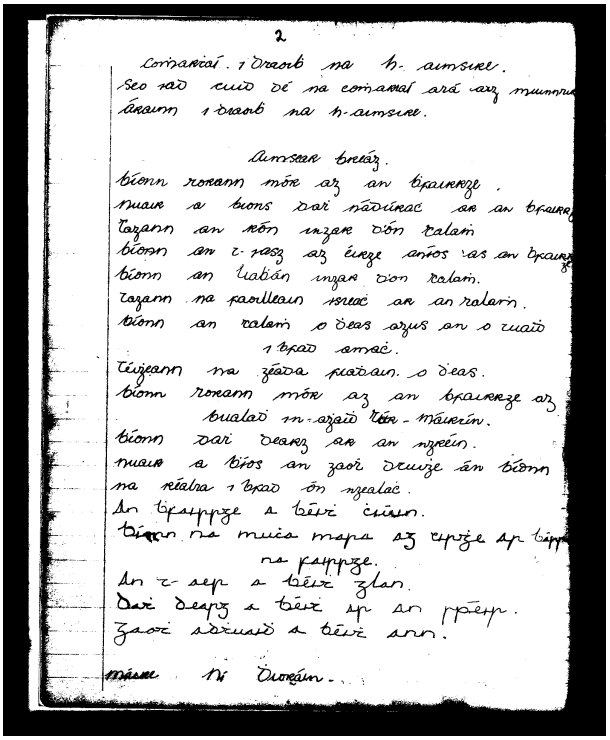
to lighter pen strokes. This method can then not be trusted as cases exist where the text on a given page is exetremly light and would not show the text in the output.

Trial and Error One other method of global thresholding is to simply manually select the threshold which is applied. This is then done on a trial and error basis and is judged by eye. The threshold is then modified by the user until an appropriate threshold is found. While this method may eventually provide a superior threshold as it requires manual intervention to select, this method makes the assumption that a single threshold will be equally suitable for all images in the dataset. Overall this method may not be worth the time required versus the moderate improvement to the overall threshold, especially if the dataset is not stable in its image brightness.

This method can provide a good result as shown in the figure above, however is extremely time-consuming due to the manual nature of the work and cannot be replicated across many people's research.



(a) Sample Page

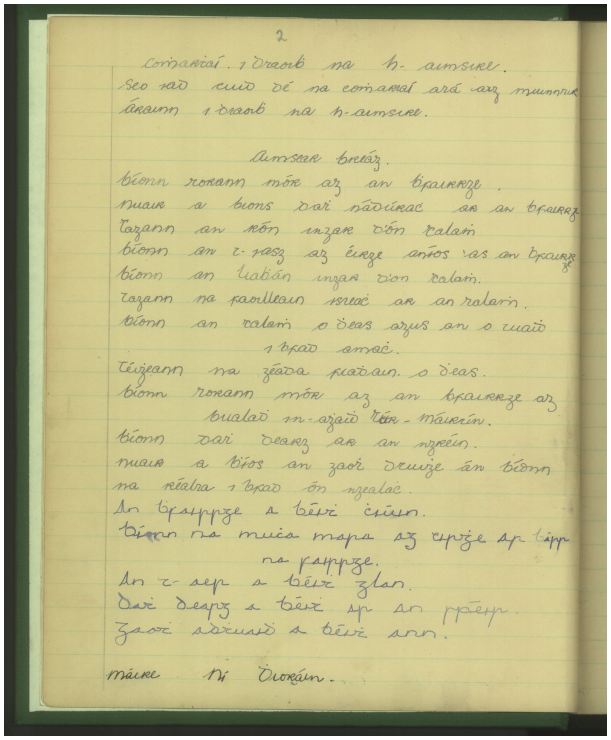


(b) Result of Trial and Error Thresholding

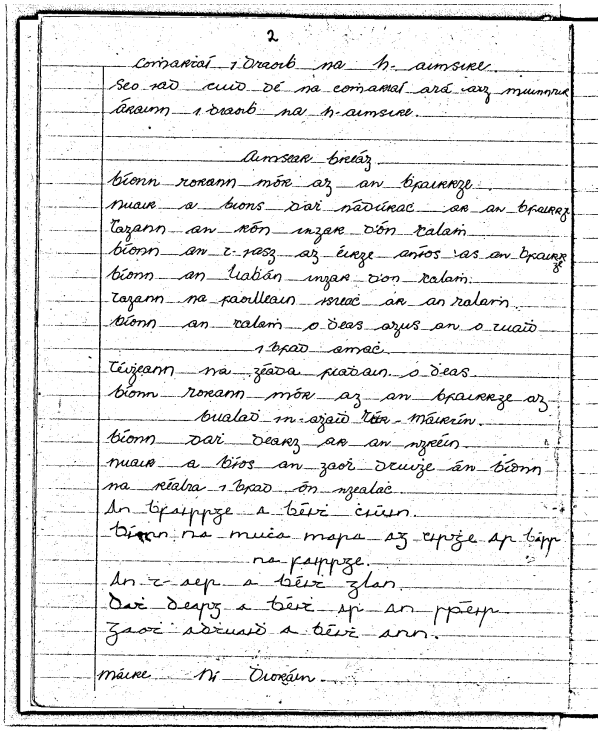
Local Thresholding

One major limitation of global thresholding is the failure to handle images which may not be bi-modal and as such have differing threshold levels of foreground and background pixels throughout different regions of the image. This limitation means that poorly lit images may suffer in quality due to certain regions being darker than others. A solution to this problem is to instead calculate an individual threshold for each pixel by inspecting the local area of the pixel and its neighbouring pixels. This allows for a change in light levels throughout the image and a change in optimal threshold. A suitably large neighbourhood must be selected as the local area must contain both a sample of foreground and background pixels in order to generate an effective threshold.

After creating examples of images using each thresholding method, it is clear that the light levels in the images are not consistent across the entire image and as such do not represent a bi-modal distribution. This creates issues when using the global thresholding methods as a single threshold is unable to accurately determine a boundary between foreground and background for the entire image. Due to this key feature in the dataset images. The choice



(a) Sample Page



(b) Result of Local Adaptive Thresholding

to use local thresholding methods was taken in order to binarize the images.

3.2.2 Noise Removal

After the binarization process, the appearance of noise is often quite common, especially in low quality or poorly lit images. The existence of noise however can be detrimental to the performance of the OCR later on as the noise may be considered as valid foreground text. An effort should be made in order to reduce the presence of noise through a variety of operations. These operations and their effects are detailed below.

note that for the following noise removal methods, the colour white is used as the colour of foreground pixels and black is used for background pixels. This colouring is reversed in the application of these techniques but does not change the underlying process by which these processes operate.

Gaussian Smoothing Gaussian smoothing is a method of reducing noise that is present in an image by blurring the image in a hope of bringing each pixel closer to its

neighbourhood's average. This method is achieved by replacing each pixel with the weighted sum of the area surrounding the pixel. This weighting is determined by a gaussian function. The function below shows a gaussian function for determining the weight for 2-Dimensional plane:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

This function can then be used to create a weighting matrix where the central index is weighted the highest and all other entries decrease the further away from the centre the entry lies. This matrix can then be used on all pixels in order to create a smoothed image. This smoothed image

Erosion Erosion is one of the two basic morphological operations which can be performed on images. Erosion involves iterating over each foreground pixel in the image with a kernel. The kernel size determines the how large an effect the operation has on the overall image. The erosion of image X by a flat structuring element B at any location (x,y) is defined as the minimum value of the image in the region coincident with B when the origin of B is at (x,y) as shown below in following equation(Ghassemian 2010).

$$[X \quad B](x, y) = \min_{(s,t)\in B} \{X(x + s, y + t)\} \tag{2}$$

Erosion has the effect of removing small clumps of foreground pixels which could be attributed to noise. This operation also thins the main foreground objects as shown in the figure below. This method can then be used in order to reduce a foreground to only its most prominent features and removing all those which are thinner than the structuring kernel.

Dilation Dilation is another of the basic morphological operations which can be performed on images and is the dual of erosion. The process itself is extremely similar as it iteratively

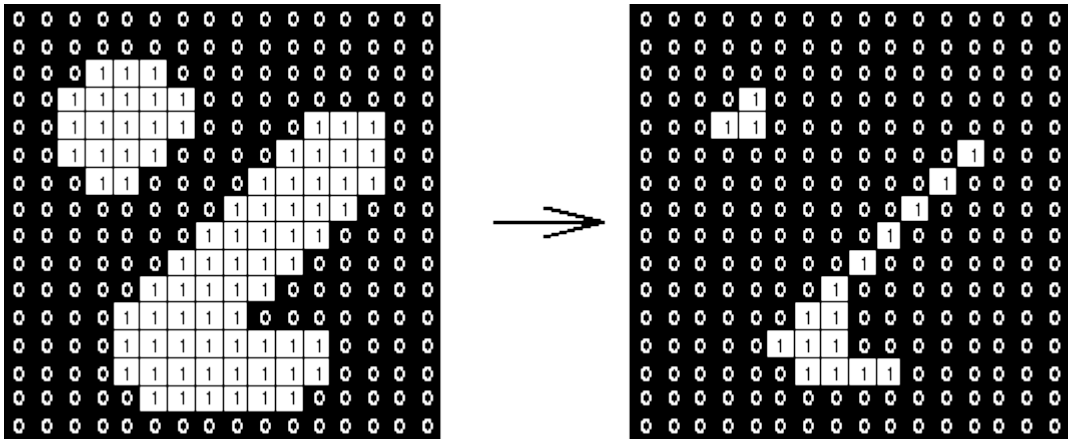


Figure 3.5: Effect of erosion using a 3x3 square structuring element. (Taken from:"https://homepages.inf.ed.ac.uk/rbf/HIPR2/figs/erodbin.gif")

positions a kernel on top of the image in order to determine whether a pixel should be changed. This process is instead defined as the maximum value of the region coincident with the structuring kernel when the origin of the kernel is at (x,y) for an image X and a structuring kernel B.

$$[X \quad B](x, y) = \max_{(s,t)\in B}\{X(x+s, y+t)\} \tag{3}$$

Dilation then has the effect of increasing the size of foreground clumps by adding the area around each clump to overall mass. One application of this effect is to close small holes which may appear within areas of foreground due to noise or other causes or to connect two foreground areas which are extremely close to one another. Note how in the figure below, the use of dilation connects the foreground clumps into one single area.

Opening Opening is a morphological operator which can be performed on images and is derived from the use of both erosion and dilation. Erosion is used in order to remove small clumps of foreground noise, however the use of erosion also affects the entire image by thinning the foreground area, reducing the overall quality. Opening rectifies this by first performing erosion on the image in order to reduce the foreground noise and then performs dilation in order to replace the eroded pixels on the non-noise foreground. This operation can then be used in order to remove small clumps of foreground noise without thinning the

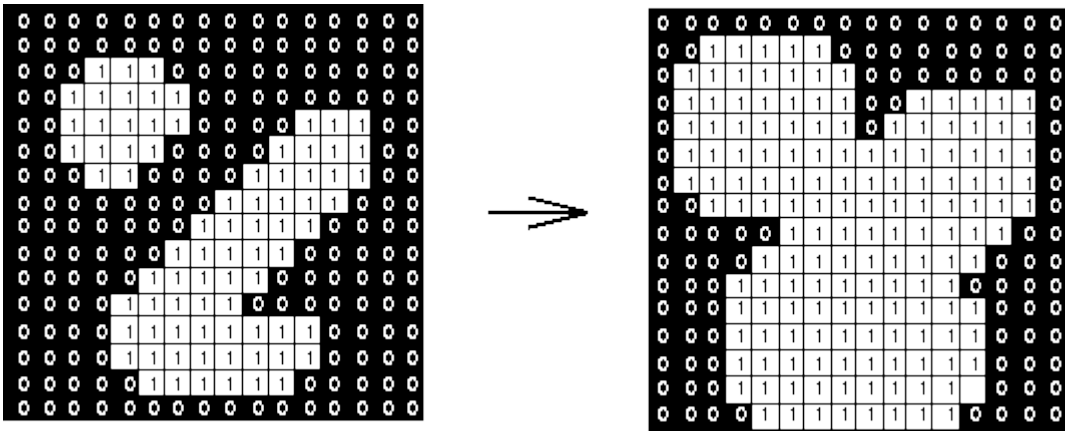


Figure 3.6: Effect of dilation using a 3x3 square structuring element. (Taken from: "https://homepages.inf.ed.ac.uk/rbf/HIPR2/figs/diltbin.gif")

overall foreground as shown in Figure 3.5.

Closing Closing is the opposite of opening and as such, is also derived from the erosion and dilation operations. This operation is used in order to fill small holes which can appear in foreground objects due to noise. This operation fills in the holes by first applying the dilation operation to the image which should fill in any holes which are present. The erosion operation is then used in order to trim the edges of the foreground back as they were also dilated. The result of this operation is that small holes in foreground objects caused by noise are filled, strengthening handwriting strokes and removing white noise from the foreground.

3.2.3 Final Preprocessing Pipeline

In order to reduce and simplify each image, a series of preprocessing operations are performed on each image. The first step in this pipeline is to binarize the image using local thresholding, this step retains all handwriting in the image. Noise is introduced throughout the image however so a series of noise removal techniques must then be employed. The closing operation is first performed the entire image, this removes a high percentage of the random noise clumps which appear throughout the image, the handwriting itself is thinned somewhat due to this image however so the opening operation is then performed in order to

enhance the thickness of the handwriting strokes. Once this process has been completed, the image is ready to be added to the training set and used in the Neural Network.

3.3 Neural Network Training

Once an image has been preprocessed, it is ready to be used in the training process itself. The pipeline is a iterative process, where the creation of training files facilitates the creation of more training files. This iterative process continues until the OCR reaches a satisfactory level of accuracy across the testing set. The training pipeline itself follows the following steps. The following pipeline follows on from previous research carried out by Zhiyuan 2017 and Rhaman 2015 by adapting and updating the pipeline for the latest version of Tesseract. Figure 3.7 shows a flow chart for the training process, starting from a fresh image and iteratively training the network in order to generate text from a source image.

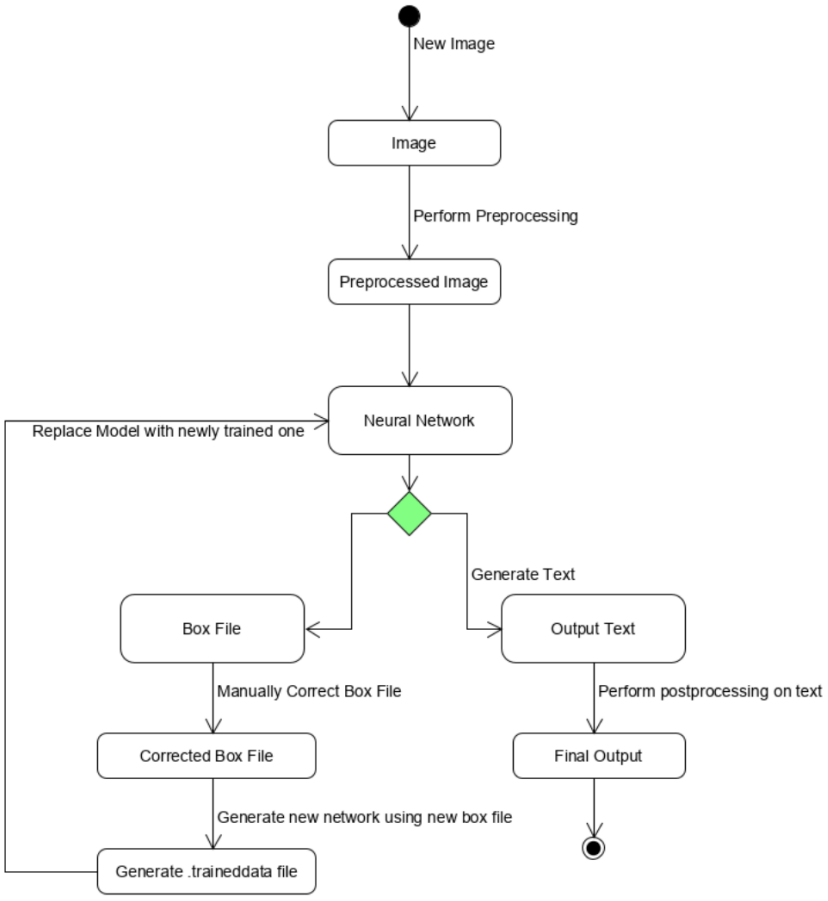


Figure 3.7: Flow chart showing the training pipeline.

1. *Generate the training image through preprocessing:* This step involves the utilisation of the preprocessing pipeline outlined above, this process involves the binarization of the source image along with several noise removal techniques in order to reduce the amount of noise introduced throughout the binarization process.

2. *Generate box files:* This step involves the creation of annotated box files which detail the location and size of all present characters in the training image. The first column denotes the character which the current line is describing, the second and third column denote the x and y pixel values of the top left corner of the bounding box, and the fourth and fifth column supply the height and width of the bounding box respectively. The last number is the page number of the image in the .tif file format. Tesseract can automatically generate these files given the training image in order to reduce the amount of time required in order to create these files. This provides virtually no advantage at the start however as the OCR has very little training data to work with and as such does not effectively recognise the characters. The following command can be used in order to generate a box file from a given .tif file; `tesseract <filename>.tif <filename> batch.nochop makebox`

3. *Manually correct box files:* Once a box file has been generated, it must be manually corrected as the network will not be able to accurately recognise and annotate the box file accordingly. As such, the correction of the box file can be extremely time-consuming as even with the help of a tool such as JTessBoxEditor, each box may take as much as 40 seconds in order to ensure the correct character has been recognised and to orientate the box correctly. As each image has on average 600-800 characters on a page, a single box file may take upwards of 5 hours to create by hand if done efficiently. The time required to complete this task can lessen over time as the model becomes more effective at recognising characters however the manual cost of ensuring 600-800 characters have been recognised correctly is still present even when the network becomes more proficient. Overall this is the most time-consuming and costly step of the entire process as is it not only integral to the entire training process but is also essential as this step ensures that the network is being trained using correct

values.

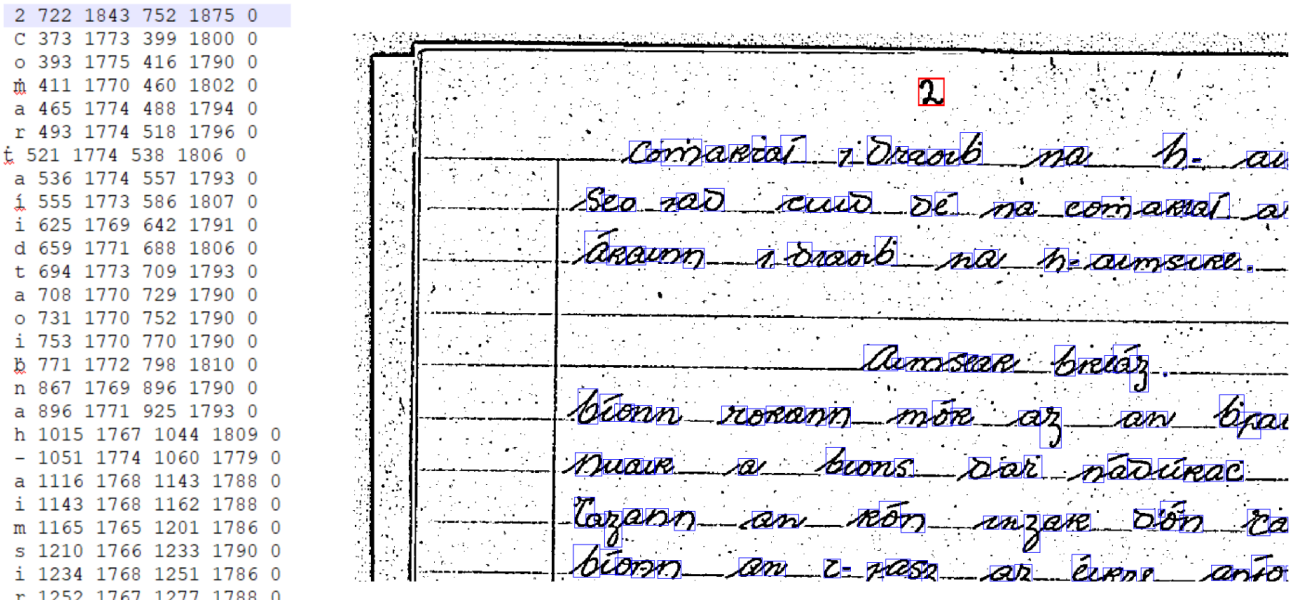


Figure 3.8: Sample of a box file pair. The left file contains a line detailing a character and a bounding box for each character present in the right file.

4. *Generate .tr files:* Once a box file has been corrected, it can be paired with the corresponding source image in order to create a training file. This file can then be grouped along with other training files to train the model. The following command is used to create the raining file. The first parameter is the name of the .tif file containing the image. The corresponding box file must then share the same filename in order to be paired correctly; `tesseract <filename>.tif <filename> nobatch box. train`

5. *Generate unicharset file:* After the training files have been created, the unicharset can be created. This file contains the full character set from the provided training files which can then be regarded as possible outputs for the OCR. The command for generating the unicharset is provided below. This command requires for each .tr training file be explicitly entered so the following command is a sample command, the actual command requires a script in order to generate due to the volume of .tr files: `unicharset extractor fontfile 1.box fontfile 2.box`

6. *Create the font properties file:* As Tesseract is primarily used in the recognition of printed text, a font file must be provided containing properties of the fonts present in

a language. The file then contains a single line for each font of the trained language where the following properties are listed in the following format: <font name> <italic> <bold> <fixed> <serif> <fraktur>, the properties are then represented by a boolean value if the variation can be present in the font. For example if a given font can have bold text, the bold property will be represented by a 1 or a 0 if bold text is not present. For the recognition of handwriting however, all of these properties are not applicable and as such, the font properties file consists of a single line: "handwriting 0 0 0 0 0"

7. *Creating a .traineddata file:* Once the unicharset has been created, a starting .traineddata file must be created in order to start the training process. this can be done with the `combine_lang_data` which takes the unicharset file and the language being trained as input in order to create a starting .traineddata file. this command can also take optional files which provide context about the language itself such as a word list file which lists all acceptable words along with a punctuation file which provides a list of acceptable punctuation.

8. *Train the network* Once all files have been created from the previous steps, the network can then be trained. The network can be trained using the lstmtraining script in the training directory of the tesseract repo. Several parameters must be supplied to the script such as the starting .traineddata file, a list of training files as well as a list of testing files, and a max iterations limit which determines at what point the training will terminate. Once the training is finished, the .traineddata file will then contain the current network model.

Once these steps have been carried out to completion the updated .traineddata file can then be used in order to generate more box files with which to train. The generated box files should then be more accurate and as such require less correction. It is this cycle which must repeat itself in order to expand the training set and improve the overall effectiveness of the model.

3.4 Evaluation of Output

Once the training has reached an acceptable level, the final model must be evaluated in order to determine its overall effectiveness at recognising characters. This method must take into account the limitations of the scraped transcriptions which will be outlined below. This final method must then provide a meaningful metric which can be used to provide a comparison of different trained models in order to determine whether the network is effective in its ability to recognise characters or can show that given more data, it can become effective at recognising characters.

3.4.1 Groundtruth Limitations

Along with the images which were used for training, a manually transcribed transcription was also scraped from the Dúchas website. This transcription was originally considered as the true groundtruth meaning that a simple string matching algorithm could be used in order to determine the error rate of specific characters. However upon further inspection of the transcriptions, several factors were identified that revealed that the transcriptions were not a true representation of the whole text present on an image but rather a human interpreted version of the text. The following errors were identified with transcriptions.

- Missing author names: At the end of a page, authors would often sign their names. These author signatures are not present in the transcription even when they are part of the main body of text on an image.
- Missing page numbers: Page numbers are often handwritten at the top of the page as the stories are written in notebooks, these numbers are not transcribed and as such can offset the transcription from the outputted result.
- Spelling corrections: Spelling errors in the image are often corrected in the transcription, these corrections can take the form of replacing characters or by inserting or deleting extra characters in order to correct a mistake. These corrections

then do not fully reflect the true text in the image.

- Missing titles: As stories contained within the notebooks often span multiple pages, the title of a given story is often written in the top margins of a page. This title is only transcribed on the first page but not continued for each page of a given story.

Due to these limitations of the collected groundtruth, any method of evaluation should not assume that the groundtruth defines the correct starting point or ending point of text present in the image. The missing title and page numbers may cause the outputted text to contain text which precedes the main body of the text which is transcribed. The missing author signatures can then cause the outputted text to have extra text at the end when compared to the transcription. All evaluation methods must then take these factors into account when determining their evaluation metric.

3.4.2 Levenshtein Distance

The Levenshtein algorithm is one of the most common algorithms to calculate the similarity or as what is known as the edit distance of two equal length strings(Shehab 2017). The distance itself is calculated as the amount of single character insertions, deletions or replacements that are required to transform one string into another. One major advantage of this metric is that the algorithm calculates the distance for each alignment possible between the two strings and outputs whichever distance is lowest. This practice thereby bypasses all misalignment issues stemming from text which is present in the true groundtruth but not in the transcription.(Gunawan 2015)

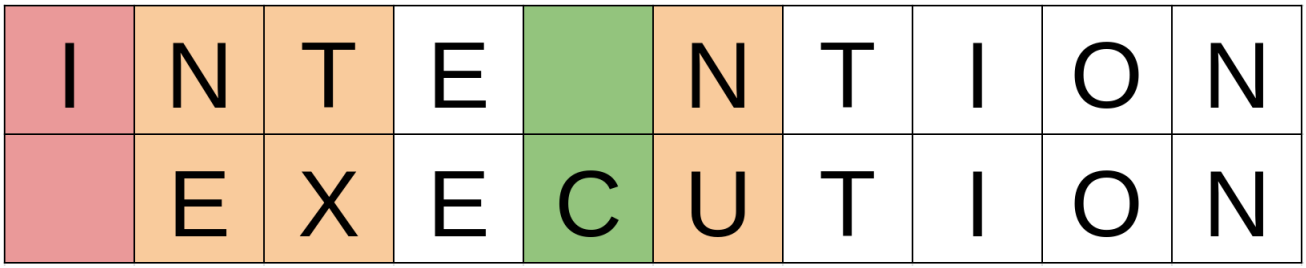


Figure 3.9: Levenshtein distance between "INTENTION" and "EXECUTION" with the insertion operation coloured green, the deletion operation coloured red and the replacement operation coloured orange.

For the purpose of comparing the output with the transcription text, a lower bound can then calculated as the edit distance between the transcription and the true groundtruth. This method can then accurately describe how close the outputted text is to the transcription without being affected by misalignment issues which can cause matching text to be incorrectly misinterpreted as incorrect.

Normalised Levenshtein Distance As the Levenshtein distance between two strings is the amount of single operations required to transform one string into another, it is naturally affected by the length of the input strings. In order to efficiently compare the levenshtein distances across many sample inputs, normalisation is required. As the transcription text is being used as the groundtruth despite the limitations mentioned above, the calculated levenshtein distances will normalised by dividing by the length of the transcription string as shown in the equation below where x is the output of the neural network and y is the transcription string.

$$NLD(x, y) = \frac{LD(x, y)}{|y|} \tag{4}$$

Following on from this equation, the average normalised levenshtein distance can be calculated across all images in the testing set in order to provide a metric which represents the network's performance across the entire testing set.

3.4.3 Testing Set Creation

In order to measure how effectively the network is recognising text, a set of test images must be collated with their respective transcriptions. This set is created from the scraped files and is separated such that no image in the testing set can be used for the training set. This ensures that the network is always being tested against unseen images and therefore is not being trained for specific scenarios. This testing set is created before the training process begins in order to ensure the prevention of leakage from the training set into the testing

set.

3.5 Evaluation Pipeline

As the normalised levenshtein distance is used as the main evaluation metric for the performance of the trained neural network, a pipeline must be created in order to determine the overall performance of network across the entire testing set. For this purpose, the average normalised levenshtein distance was used. This metric was calculated by first using the the trained network to output a text file with its transcription of a given image. The normalised levenshtein distance was then calculated using the outputted transcription and the source transcription. This step is then repeated for each image in the testing set and calculating the average normalised levenshtein distance across the entire testing set. This metric can then be used in order to compare differently trained networks with one another.

4 Results and Discussion

Following the evaluation method outlined in the previous chapter, the performance of the neural network can be tracked over successive training iterations. This tracking provides insight on the performance of the neural networks as the training set grows. These results can then help determine the relationship between the performance of the neural network across several factors. By examining the changes in the performance of the neural network, the applicability of the implementation can be determined. This chapter will present and compare the performance of the trained neural network against certain metrics and discuss the conclusions which can be derived from these comparisons.

4.1 Results

As mentioned previously, an aim of this dissertation is to determine whether the proposed approach is suitable for further research. In order to determine this, the current progress of the training must be tracked so that the effect of future work may be estimated.

Figure 4.1 compares the average normalised levenshtein distance against the amount of words used in the training set. This comparison can be used in order to estimate the overall performance of the network based on the amount of training data used in its creation and the change in performance as that amount of training data increases.

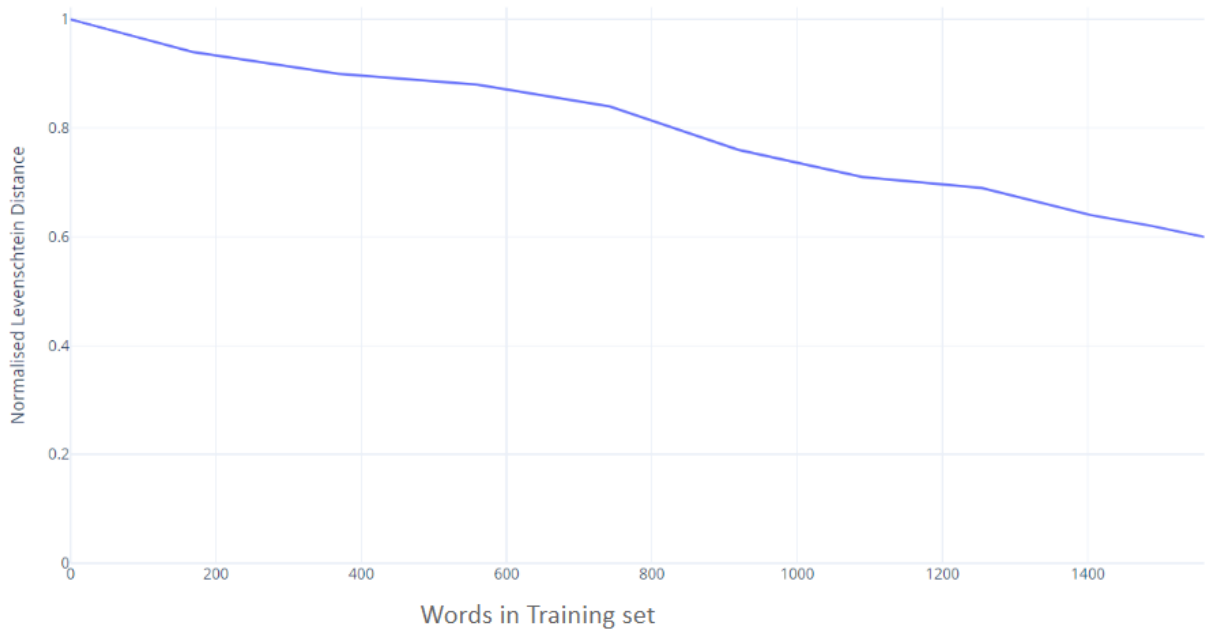


Figure 4.1: Comparison of the average normalised levenshtein distance against the amount of words used to train the neural network

4.2 Analysis of Results

Through analysis of this result, it is clear that the average normalised levenshtein distance decreases as the amount of training data increases. This result shows that the overall performance of the network itself increases along with the training data. While the amount of training data available was constrained for the purposes of this research due to time constraints, the effect of an increased training set on the performance could be a potential area for future research.

5 Conclusion

This chapter reviews the dissertation as a whole with reference to the research aims defined in the introduction. The chapter then concludes the dissertation with thoughts on future work which could take place in the field.

5.1 Assessment of Research Aims

After defining a set of aims and goals in the introduction to this dissertation, this section will now review the research done in order to determine whether the goals were sufficiently met.

In relation to the primary aim of the dissertation, the implementation section successfully explores the process in creating a neural network based approach in order to tackle the problem of recognising handwritten traditional Irish script. The feasibility of this approach is then tested using the evaluation method outlined in the section 3.5. Overall this approach demonstrates that a relationship exists between the performance of the neural network and the size of the training set used.

The secondary aim of creating a pipeline for training and evaluation purposes is again outlined in the implementation which details the steps required in order to create an iterative training process along with a method of evaluating the overall performance of the trained model on a defined testing set.

5.2 Final Thoughts

Neural networks have an extremely broad range of use. This range is constantly being explored and expanded with the publication of new research detailing new uses or applications. This research is one of such publications, detailing the potential use of neural networks in a previously unexplored field of research. The results of this research showed that while it did not meet the industry standard, the performance of the model itself is linked to the size of the training set which could be expanded upon in future works. Overall the research shows that applications do exist for neural networks in the field of handwritten traditional Irish script recognition.

Bibliography

H.T. Siegelmann, E.D. Sontag (1995). "On the Computational Power of Neural Nets". In: *Journal of Computer and System Sciences*.

Sepp Hochreiter, Jürgen Schmidhuber (1997). "Long Short-Term Memory". In:

Zhang, G.P (2000). "Neural networks for classification: a survey". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*.

Kay, Anthony (2007). "Tesseract: an Open-Source Optical Character Recognition Engine". In:

Ghassemian, M. Khodadadzadeh ; R. Rajabi ; H. (2010). "Combination of region-based and pixel-based hyperspectral image classification using erosion technique and MRF model". In: *2010 18th Iranian Conference on Electrical Engineering*.

Tan, Bolan Su ; Shijian Lu ; Chew Lim (2011). "Combination of Document Image Binarization Techniques". In: *2011 International Conference on Document Analysis and Recognition*.

Gunawan, H.A. Maarif ; R. Akmeliawati ; Z.Z. Htike ; Teddy S. (2015). "Complexity Algorithm Analysis for Edit Distance". In: *2014 International Conference on Computer and Communication Engineering*.

Rhaman, Muhammed Tawfiq Chowdhury ; Md. Saiful Islam ; Baijed Hossain Bipul ; Md. Khalilur (2015). "Implementation of an Optical Character Reader (OCR) for Bengali language". In: *2015 International Conference on Data and Software Engineering (ICoDSE)*.

Sexton, Conor (2017). "Advancing Neural Turing Machines:Learning a Solution to the Shortest Path Problem". MA thesis. Trinity College Dublin.

Shehab, Khaled Balhaf ; Mohammad A. Alsmirat ; Mahmoud Al-Ayyoub ; Yaser Jararweh ; Mohammed A. (2017). "Accelerating Levenshtein and Damerau edit distance algorithms using GPU with unified memory". In: *2017 8th International Conference on Information and Communication Systems (ICICS)*.

Zhiyuan, Liu Shufeng; Shen Shaohong; Sun (2017). "Research on Chinese characters recognition in complex background images". In: *Image Vision and Computing (ICIVC) 2017 2nd International Conference*.

Khan, Md. Abu Bakr Siddique ; Rezoana Bente Arif ; Mohammad Mahmudur Rahman (2018). "Digital Image Segmentation in Matlab: A Brief Study on OTSU's Image Thresholding". In: *2018 International Conference on Innovation in Engineering and Technology (ICIET)*.