The University of Brighton

A dissertation presented for the degree of
BSc (Hons) Computer Science

Using Machine Learning to Recognise Patterns in the Application of Optical Character Recognition

George William Lancaster
15808173

Supervised by Dr. Stelios Kapetanakis

May 7, 2018

Contents

List of Figures

List of Tables

Abstract

This project explores the different techniques employed in image recognition, covering topics such as feature extraction, computer vision and machine learning. Through research and independent testing, the best performing image recognition techniques were identified and applied in the construction of a classifier capable of recognising hand-written text. This has been used in conjunction with a feature extraction algorithm to produce a web application that can detect and classify handwriting from an image. In testing, the best performing classifier achieved a percentage error of 4.99 when classifying individual letters from the EMNIST-letters data set. The web application has an approximate percentage error of 3.70 when tested on perfect data.

Chapter 1

Introduction

Machine learning is defined as "the field of computer science that gives computers the ability to learn without being explicitly programmed" [Samuel, 1959]. Today, machine learning is arguably the fastest growing area of computer science, and is currently used for technologies such as drug discovery, autonomous vehicles and natural language processing.

Optical Character Recognition (OCR) is a rapidly developing field within machine learning. Its conception dates back as far as 1870 with Charles R. Carey's invention of the retina scanner [Schantz, 1982]. Due to its unpredictable nature, there is no common, repeatable solution for classifying handwriting with 100 per cent accuracy. Therefore, the classification of hand-written characters is still considered to be an open research topic [Diem *et al.*, 2013].

1.1 Aims and Objectives

The aim of this project is to demonstrate an understanding of machine learning through performing various experiments on neural networks. These include measuring how the length of training, type of training data, and use of ensembles of successful classifiers affects classification accuracy. The goal of this is to apply an intelligent classifier to a web application capable of recognising handwriting.

The following requirements were specified in the interim planning and investigation report:

1. Relevant machine learning techniques must be explored, analysed and experimented with;
2. The deliverable must be able to classify hand-written numbers with a percentage error of 30 to be considered successful;
3. The deliverable must be able to classify upper case hand-written letters with a percentage error of 30 per cent to be considered successful;

1.2 Stakeholders

The stakeholder in this project is Stelios Kapetanakis, the project supervisor.

1.3 Scope

The project is split into four sections; performing research into the optimisation of convolutional neural networks, applying the research in classifier training, the development of a feature extraction algorithm and the application of the trained classifiers to an optical character recognition web application. The scope of this project is open, as improvements can always be made to the accuracy of classifiers.

Some techniques that have been used to test optimisation include:

1. Measuring how the length of training affects accuracy;
2. Measuring how applying transformations to training data affects accuracy;
3. Measuring how using collections of classifiers affect accuracy.

Three classifiers have been deployed in a web application, which has the ability to upload photographs to a canvas, or draw text using a mouse. Users can create an account to save and download their classification history, or continue as a guest.

The application can classify upper/lowercase letters, numbers, or upper/lowercase letters and numbers.

1.4 Modules Used

This project expands upon the modules:

1. CI231 - Intelligent Systems;
2. CI284 - Data Structures and Algorithms;
3. CI346 - Programming Languages, Concurrency and Client-Server Computing;
4. CI222 - Project Planning and Control.

1.5 Summary of Outcomes

The best performing classifier achieved a 4.99 percentage error when tested against the EMNIST-letters data set. It was created using an ensemble of what had previously been established to be the best performing classifiers. When tested on data extracted using the document analysis algorithm, this classifier achieved a 3.70 percentage error; this is an estimate to the accuracy of the web application.

The digit classifier achieved a percentage error of 0.29 when tested against the EMNIST-digits data set, and the universal classifier has a 14.15 percentage error when tested against the EMNIST-byclass data set. Both of these networks were trained using rotate and zoom transformations.

Chapter 2

Background

2.1 Tools and APIs

The machine learning algorithms have been implemented in Python using Keras with a TensorFlow backend. A full list of the packages used can be found in the project folder in appendix D. If running the application on a machine without an NVIDIA graphics card with CUDA cores, the *requirements-cpu.txt* must be used. Otherwise, *requirements-gpu.txt* can be used for faster training times.

2.1.1 TensorFlow

TensorFlow is an open-source machine learning framework. It has rapidly become the industry leader in the production of machine learning models [Abadi *et al.*, 2016].

2.1.2 Keras

Keras is a high-level neural network API that acts as a wrapper for TensorFlow [Chollet, 2018]. It has several methods for creating, training and testing neural networks, and allows for classifiers to be defined in layers. Keras was chosen for the machine learning aspect of this project, as it is the most user-friendly API available whilst still using the full power of TensorFlow. This is the most important factor, given the limited time to complete this project.

2.1.3 OpenCV

OpenCV (Open Source Computer Vision Library) is a library written in C/C++. Due to its popularity, wrappers for other programming languages have been developed to increase its usability, including Python [Bradski and Kaehler, 2000]. OpenCV has been used in this project to detect classifiable features in photographs, and for the various image manipulation functions it offers.

2.1.4 Graphical Processing Unit

The training of a machine learning classifier can be accelerated using a Graphical Processing Unit (GPU) rather than a Central Processing Unit (CPU). Training with a GPU tends to be faster as they are made up of many simple cores and can have thousands of concurrent hardware threads. In contrast, CPUs have few complex cores and are limited to single-threaded performance [Steinkraus *et al.*, 2005]. For this project, an NVIDIA GeForce GTX1050ti was used to speed up training. Initial testing showed that using a GPU sped up training time by over three times, shown in appendix E.

2.2 Computer Vision

Humans use sight to view the world with ease, we can detect objects and process visual information autonomically [Szeliski, 2010]. When looking at an object, light hits the light-sensitive retina in the eye. This transmits a signal to the brain which processes an image. Current approaches to computer vision mimic this process, replacing the organic components with digital ones [Boyle and Thomas, 1988]. The eye can be replaced with a piece of hardware, such as a digital camera or scanner. Software is used to recognise objects from within an image.

Computers render still images in two formats; vector and raster. Vector images can be thought of as a mathematical equation containing points, lines and curves related to one another. These are usually graphics and are not generally used for digital photographs and recordings [Kaufman, 1993]. Computer vision tasks can not currently be used on these types of images unless converted to a raster format. A raster image is a grid of x and y coordinates on a display space, with each coordinate directly mapping to a pixel value [Hughes and Foley, 2014]. At an abstract level, a raster image is just a method of displaying numerical data. When we enlarge a part of an image, we can see the individual pixels and how they collaborate to form a larger image.

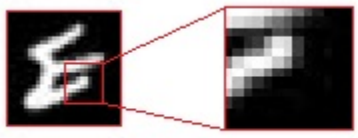


Figure 2.1: Two views of a raster image taken from the EMNIST data set. As the size of the image increases, each pixel becomes more pronounced.

Computer vision is a branch of artificial intelligence that analyses images, with the aim of gaining high-level understanding. Currently, machine learning techniques using artificial neural networks (ANN), are generally accepted to have the highest success rate at this kind of classification task [LeCun *et al.*, 1998a].

2.3 Machine Learning

The goal of machine learning is to create a function, or classifier (f), to predict an output (Y) based on an input (x) using the same formula as any other function.

$$Y = f(x) \tag{2.1}$$

Firstly, the classifier f is randomly initialised so that the function gives a random output. To create a more accurate function, it must be trained on a set of data [Rogers and Girolami, 2016].

Labelled training data can be used to supervise learning, by using pairs of inputs and their output labels to build the function using backward propagation. The function will learn specific features of an input, and how to map it to a label. The ultimate goal of training is to create a function strong enough to classify new, unseen data.

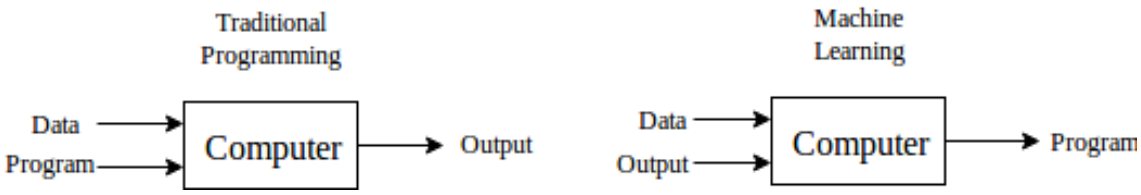


Figure 2.2: A diagram showing the differences between traditional programming and machine learning.

2.4 Neural Networks

A neural network is a data processing model inspired by the biological construct of the brain. It is comprised of processing units called neurons, modelled and named after the neurons found in the brain [Goodfellow *et al.*, 2016].

2.4.1 Neurons

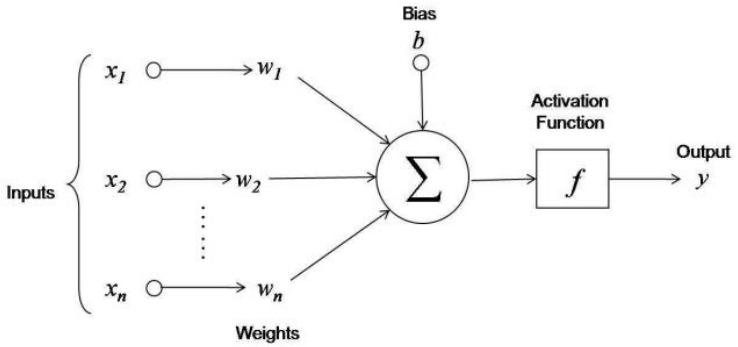


Figure 2.3: A visual representation of a neuron [Cornell, 2018].

Fig 2.3 shows that a neuron has many inputs $x_1, x_2, ..., x_n$, each connected to a weight. The weight of an input defines its importance using a real number. The higher the number of the weight, the larger the effect it has on the output. The bias (b) is a metric that defines the difficulty of neuron activation. A neuron with a larger bias will activate more easily, whereas a negative bias means that it is more difficult. During training, we can change the weight and bias values using the backward propagation algorithm to allow different neuron activations depending on the input.

The activation function checks the neuron's output to ensure that the value is significant enough to warrant its activation [Nielsen, 2015]. In its simplest form, the neuron will activate if its calculated input is positive, shown by the function:

$$output = \begin{cases} 0 \; if \; w \cdot x + b \leq 0 \\ 1 \; if \; w \cdot x + b > 0 \end{cases} \tag{2.2}$$

This describes a step function. One of the most popular variations of this is the Rectified Linear Unit (ReLU) activation function, which returns 0 if it receives any negative input. For any positive value ($w \cdot x + b$), it returns the input back [Glorot *et al.*, 2011].

Activation functions in neurons are not limited to the ReLU function, with other options including the Sigmoid and Tanh functions. However, the ReLU function is currently considered to be the most effective [Ramachandran *et al.*, 2018]. One advantage of using the ReLU activation function is that upon random initialisation, only half of the units will be activated. This results in a sparsely activated network that is more sensitive to training. Another advantage

of the ReLU function is that it is efficient to compute, as it only uses addition, multiplication and comparison. This means that it has a minimal effect on the speed of training.

Although the ReLU function is highly efficient, it is not without its drawbacks. Some neurons can 'die' through having their parameters pushed so low that the neuron can not be activated [Maas *et al.*, 2013]. This is generally caused by having the networks learning rate set too high.

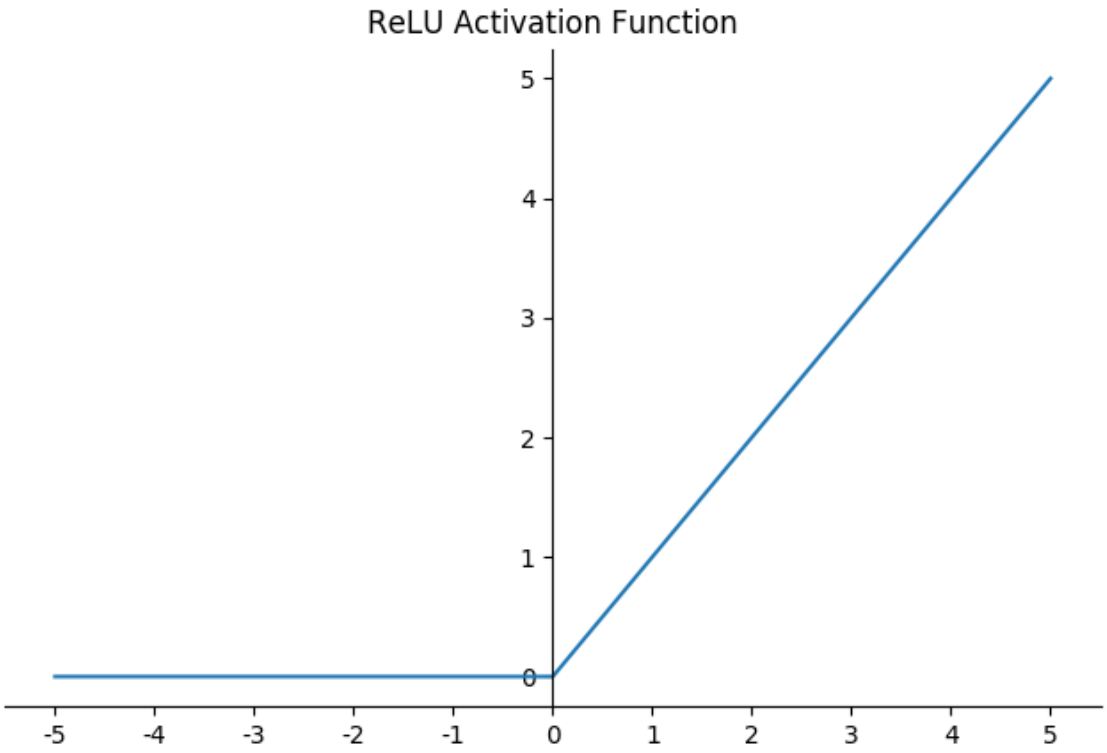


Figure 2.4: A visual representation of the Rectified Linear Unit function. When x reaches 0, $y = x$.

2.4.2 Layering

Each neuron represents a decision. Through building layers of interconnected neurons, decisions can be made on increasingly complex tasks by using the output of one neuron as the input for another, generating higher levels of abstraction.

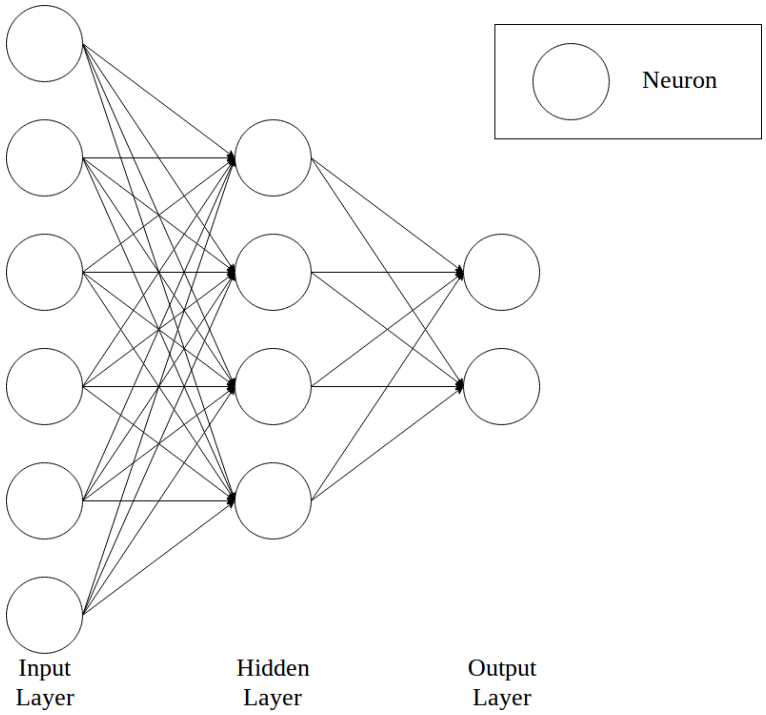


Figure 2.5: A visual representation of a simple, fully connected neural network model. It is fully connected because all neurons from one layer are connected to all neurons on the next layer.

The most powerful ANNs are constructed of at least three layers [Smith, 2013]:

1. Input layer - This layer takes an input vector in the shape of the number neurons in the input layer. This is a passive layer that does not modify the input data.
2. Hidden layer(s) - These layers do not directly deal with the input or output of the network. They can be connected to one another, and to the input or output layers. The hidden layers perform the main computations through the connection of neurons. Powerful ANNs have many hidden layers.
3. Output layer - The output layer processes the result of computations. Usually, this is to normalise the values using a function.

2.4.3 Backward Propagation

The backward propagation (backpropagation) algorithm is used to adjust the weights and biases of the neurons during training, minimising the error between the predicted and expected output [Carling, 1992]. Backpropagation is a two-stage process that uses a forward and backward pass. The forward pass makes a prediction on a data item. If the result is incorrect, the network

performs the same process in reverse, however it has both the predicted output and expected output to make adjustments to weights, bringing the predicted result closer to its expected value [LeCun *et al.*, 1998b]. This is done by measuring the effect changing the weights has on the loss function. A lower loss function indicates that a classifier is learning the training data, however it can not detect when a classifier is over or underfitting.

2.4.4 Loss and Optimisation

A loss function is used to calculate a number that suggests how wrong a prediction is. Cross-entropy is a commonly used loss function that scales based on the correctness of the neurons output. In effect, the larger the error in the output, the larger the value calculated by the cross-entropy function [Nielsen, 2015].

The goal of optimisation is to change the weights of the neurons to minimise the loss function, and therefore minimise the error in the network. One optimisation function described in a 2012 study is Adadelta, which has also been used in this project [Zeiler, 2012]. The main benefit of using loss and optimisation functions is that they allow for an adaptive learning rate. This means that the changes to the networks weights are affected by the progress of training, which results in faster and more efficient training.

2.4.5 Softmax Function

To retrieve a value from a list of predictions, the Softmax function is applied in the output layer of the network. It is a squashing function that normalises a k-dimensional array $\mathbf{z}$ of arbitrary values to values between 0 and 1, $\sigma(\mathbf{z})$. The sum all all values in the array must also equal 1 [Christopher, 2016]. The Softmax function is given as:

$$\begin{aligned} &\sigma : \mathbb{R}^K \rightarrow (0,1)^K \\ &\sigma(\mathbf{z})_j = \frac{e^{zk}}{\sum_{k=1}^{K} e^{zk}} \textbf{ for } j = 1, ..., k \end{aligned} \tag{2.3}$$

2.4.6 Underfitting

Underfitting occurs when a model is not trained on enough samples to learn the characteristics of the data set. This causes it to fail during validation as it cannot apply what it has learned to new data [Haykin, 2004].

2.4.7 Overfitting

Overfitting refers to a model that learns the training data too strictly. Rather than learning the characteristics of each class, the model learns how to map each piece of training data to its label [Haykin, 2004]. When tested on new data, it is unable to make reliable predictions. The existence of overfitting implies that the duration of training does not directly correlate with higher accuracy.

2.5 Convolutional Neural Networks

This project focuses on the use of convolutional neural networks (CNN), a type of network that uses a combination of unique layers for visual analysis. The CNN is modelled on the biological construct of the visual cortex [Hubel and Wiesel, 1968].

CNNs use kernel convolutions to extract significant features from images. A kernel is a matrix of values that can be applied to an image using convolution, which involves applying the kernel to equally sized pieces of the image to generate different effects.

In applying a kernel (K) to image section (I), in the resulting matrix (R) position $[2, 2]$ is found using:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) \tag{2.4}$$

$$R\begin{bmatrix} 2, & 2 \end{bmatrix} = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9)$$

Types of kernel convolution include: sobel operator and gradient detection. Each type uses a different kernel to accentuate features within the image. For example, the sobel operator is used to detect edges, and the gradient detection kernel detects lines of a certain gradient [Goodfellow *et al.*, 2016]. This is especially useful in image recognition tasks, as a neural network can learn the features accentuated by kernel convolution. Each image generated by the convolutional layer is known as a feature map.

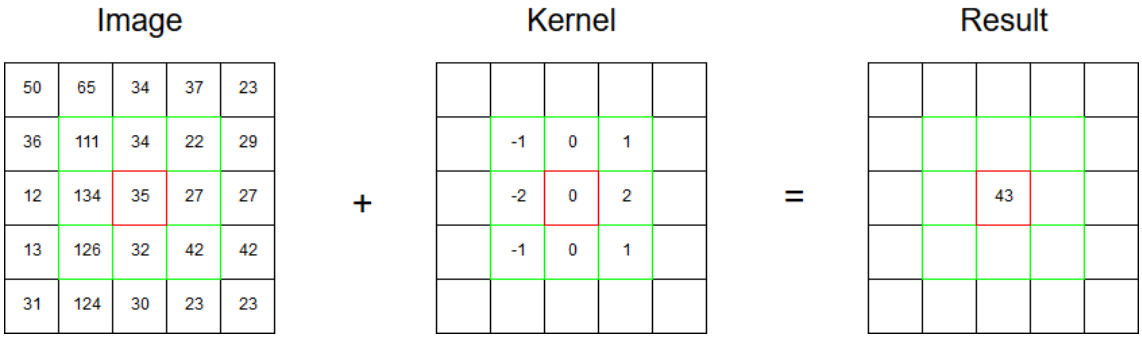


Figure 2.6: An example of the vertical-right sobel operator applied to a section of an image.

Denoting the k^{th} feature map at a given layer as h^k, with filters determined by weights (W^k) and bias (b_k), the feature map h^k is obtained using the formula [Goodfellow *et al.*, 2016]:

$$h_{ij}^k = tanh((W^k \cdot x)_{ij} + b_k) \tag{2.5}$$

Feature maps are stacked within the layers of the network so that each layer is comprised of many feature maps, produced using different kernel convolutions. This has the effect of each feature map in the layer highlighting different distinct features of an image, which can then be learned.

CNNs use pooling layers in conjunction with convolutional layers to down-sample feature maps. A matrix (M) of size (x, x) is passed over an image with a stride of x. This is done to generate a smaller image, comprised of the most prominent features of the input image. Two techniques are commonly used for pooling layers; average-pooling and max-pooling. Average-pooling calculates the average pixel within M, and max-pooling takes the largest pixel value in M. Advantages of pooling include the reduction of complex computation in the higher layers, highlighting the prominent features of the image, and most importantly adding invariance to the convolutional layers [Scherer *et al.*, 2010]. Fig 2.7 shows the differences between average and max-pooling.

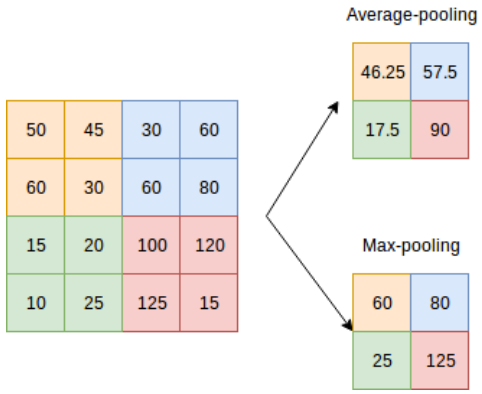


Figure 2.7: A visual representation of a pooling layer of size 2x2. The output image generated in both instances is half the size of the input image.

CNNs do not rely on fully connected layers for computation within the hidden layer, but use sparsely connected layers to control and localise the flow of data from one layer to the next [Kaiming, 2015]. This is visualised in Fig 2.8. The final layers are fully connected, and work much like layers in a regular neural network.

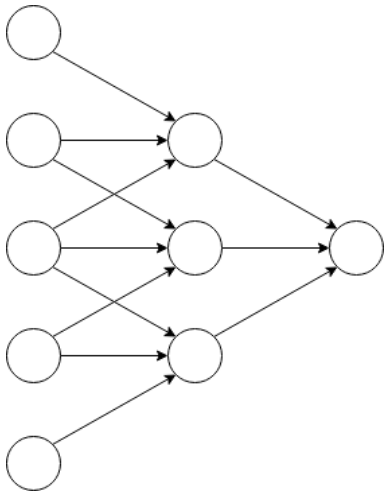


Figure 2.8: A visual representation of a sparsely connected network; not all neurons are connected to those of the next layer.

2.6 Data and Image Representation

The data set used in this project to train and test classifiers is the Extended Modified National Institute of Standards and Technology (EMNIST) database [Cohen *et al.*, 2017]. It contains 814,255 hand-written characters consisting of all upper and lowercase letters found in the English language, as well as digits 0-9. EMNIST has been chosen as it is the largest data set of its kind, and it has had success in previous studies [Cohen *et al.*, 2017] [Singh *et al.*, 2017]. Each item in the data set is a 28x28 raster image of a single hand-written character. The data is split into 5 sub-sets:

Dataset	Chars	Classes	Type
ByClass	814,255	62	Letters, Digits
ByMerge	814,255	47	Letters, Digits
Balanced[1]	131,600	47	Letters, Digits
Letters[1]	145,600	26	Letters
Digits[1]	280,000	10	Digits
MNIST[1]	70,000	10	Digits

Table 2.1: Table showing the different data sets within EMNIST.

Within Python, each image is stored as a 2-dimensional array using the Numpy library. Each pixel in the image is stored as a normalised floating-point value between 0 and 1. Its position in the array dictates its position in the image. As the images are greyscale, they have a depth of 1.

```
>>> image = test_img[1]          # read an image from the test dataset
>>> image.shape                  # get the shape of the image array
(28, 28, 1)
>>> image[15,15,0]               # get the value of the pixel (15,15)
0.14901961
>>>
```

Figure 2.9: The shape of an image from the EMNIST data set in Python.

The images within the data set use a black background with white lettering, the inverse to how we usually view writing. This is done as the pixel value 0 is black and any shade off black will be greater than 0. This means the more distinctive the feature, the closer it will be to 1.

2.6.1 Project Data Set (PDS)

In addition to EMNIST, a data set has been created called the Project Data Set (PDS). It has been created to have similar characteristics to the EMNIST data set. PDS was collected by having two participants write upper and lower-case letters onto a piece of lined paper. A photograph of the paper was taken and a data set was generated using the feature extraction algorithm found in *char_extractor.py*, the same algorithm used in the web application. Any false

[1]Balanced - each class contains the same number of characters.

detections were removed from the data set by hand. In total, there are 757 images spanning 26 classes. They are greyscale images of size 28x28 so that they fit the input layer of a network trained using EMNIST. Upper and lower case letters are contained within the same class, for example p is classified the same as P.

Data Set	Character	Graphical Representation
EMNIST	N	
EMNIST	f	
EMNIST	A	
EMNIST	6	
EMNIST	R	
EMNIST	u	
PDS	Z	
PDS	A	
PDS	O	
PDS	N	
PDS	P	
PDS	Q	

Table 2.2: Table showing a random selection of images from the EMNIST and PDS data sets.

Chapter 3

Implementation

The implementation of this project was split into four sub-tasks: feature extraction, classifier training, classifier evaluation and application integration.

3.1 Feature Extraction and Image Pre-processing

When classifying more than one character from an image, it is necessary to extract each character and convert it to a format in the same shape as the input layer of the classifier. In this case the first layer of the neural network takes an input vector of shape (28x28x1), the same shape as the training data. To do this, an algorithm has been devised within the module *char_extractor.py*, which extracts classifiable data from an input image, returning a list of images ready for classification.

3.1.1 Binarisation and Removal of Noise

Firstly, an image must have any background noise removed and its prominent features accentuated. This is done to stop any irregularities in the image from being detected, as well as ensuring that the features that are supposed to be detected, are. This is achieved using pixel thresholding to remove noise and binarise the image, and feature dilation to increase the prominence of the remaining features.

The other operation applied at this stage is a total rotation of 5° clockwise. This is done to ensure the sorting algorithm applied in later stages preserves the line formatting of each detected feature.

3.1.2 Feature Detection

OpenCV has provided a function $findContours$, which uses an adapted version of an algorithm first described in a 1985 paper [Suzuki *et al.*, 1985]. The algorithm scans an image, interrupting

the scan when an edge has been detected. In a binary image, an edge can be defined as any collection of adjoining pixels with values greater than 0 [Marr and Hildreth, 1980]. Although binarasation is not necessary for edge detection, it greatly increases the probability of finding an edge, as an edge is any non-zero value [Canny, 1987]. Through following the border of the first detected edge, an outline of a shape can be mapped and added to a list of detected features.

All images analysed by this algorithm are resized, and any features with a total area of less than 200 pixels, and with a width greater than double its height are removed. Through trial and error, these values were found to retain the important features, whilst removing false detections. The deletion of small features removes any debris, not removed by thresholding in the previous step.

3.1.3 Sorting Detected Features

As this project focuses on the English language, which reads left-to-right, the list of detected features must be sorted left-to-right, top-to-bottom. This ensures that features are classified in read-order, and therefore output in read-order.

Each detected feature is stored as a tuple value in the format (x, y, w, h), whereby $x = x$-axis, $y = y$-axis, $w =$ width, $h =$ height. This makes it possible to sort the features into read-order by their x, y values. To begin with, the features are sorted by their y values to organise them into lines. The top of one feature being below the bottom of the previous one signifies the start of a new line. Each line can then be sorted by its features x value. Whilst flaws have been observed when applying this algorithm to complicated documents, it is sufficient for the intended used of this project.

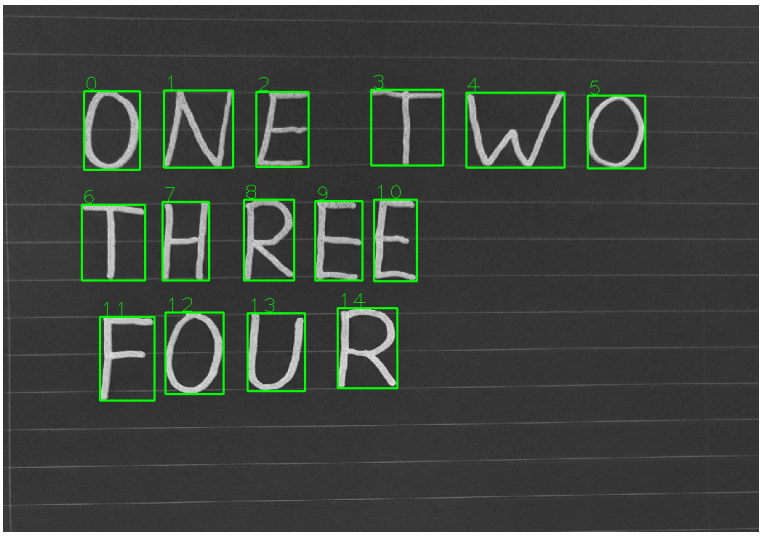


Figure 3.1: A list of detected features sorted into read-order.

3.1.4 Input Shape

To fit the input layer of the classifier, all detected features are resized so that their smallest edge (w or h) is equal to 28 pixels, done so that they are reduced in size whilst preserving their aspect ratio. The features are then further reduced in size by 30 per cent so that they can be superposed onto the centre of a white background of size 28x28. This prevents any of the features from being cropped, and keeps the detected feature centered. Each pixel is then inverted, generating an image with a black background with white feature, ready for classification.

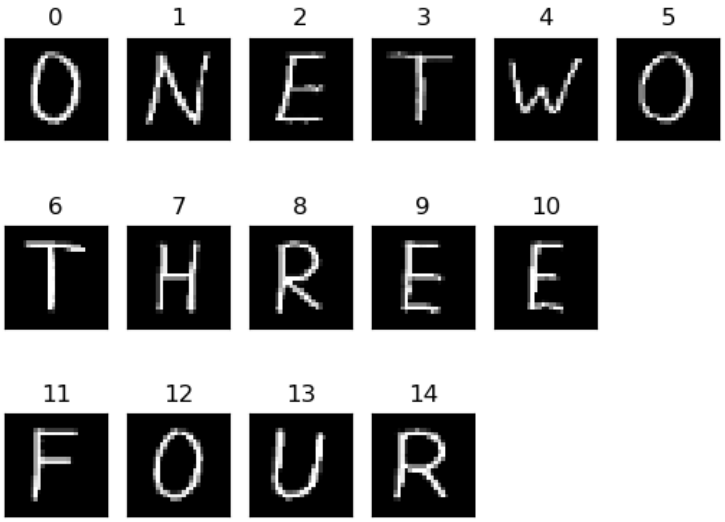


Figure 3.2: List of detected features post-processing.

3.2 Network Architecture

3.2.1 7-Layer Convolutional Network

The network used in this project (Projnet) was adapted from a neural network first described in a 1998 study, in which the performance of various machine learning classifiers had been measured in the task of classifying hand-written digits [LeCun *et al.*, 1998a]. To extend upon this study, a configuration similar to that of the most successful network 'Lenet-5' has been recreated and trained using data from the EMNIST data set. It has been implemented in this project in the module *train_network.py* function *create_lenet_5()*.

Projnet differs from Lenet-5 in the number of feature maps it produces in each layer. The original Lenet-5 network produced 6 feature maps at layer C1. Since computational power has increased since that study, this network uses 12 feature maps in layer C1, and therefore double the number of feature maps in each layer.

Fig 3.3 shows that the network spans 7 layers: C1, P2, C3, P4, C5, F6, and Output.

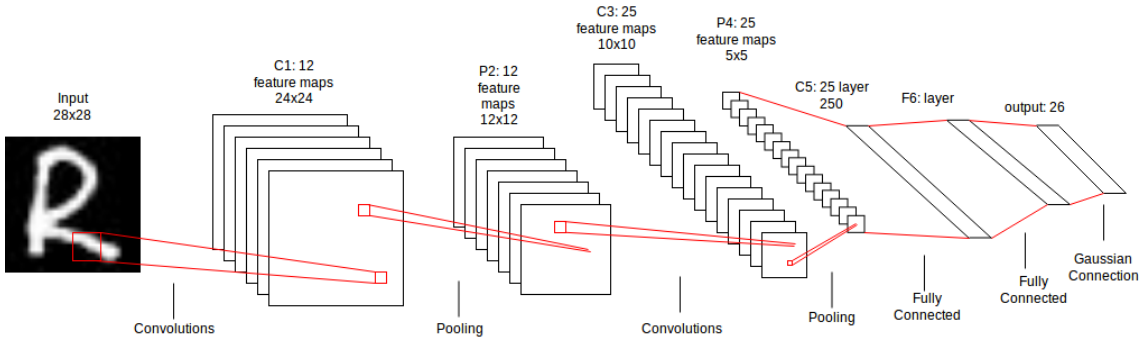


Figure 3.3: The Projnet architecture. An adaptation of the architectural diagram found in [LeCun *et al.*, 1998a], showing the flow of data through the network.

1. C1 - Applies spatial convolution to split the input image into 12 feature maps of size 24x24. The kernel size used is 5x5 and therefore 4 pixels are lost in the convolutional procedure.
2. P2 - Subsamples of each feature map are taken using a kernel of size 2x2 with a stride of 2. This layer uses max-pooling as the sub-sampling method to generate 12 feature maps half the size of the input feature maps generated in C1 12x12.
3. C3 - Applies spatial convolution to generate 25 feature maps of size 10x10. 2 pixels are lost due to the kernel size being 5x5.
4. P4 - This layer generates an output of 25 feature maps of size 5x5.
5. C5, F6 - Flatten the 3D output to a 1D tensor using fully connected layers.
6. Output - Reduce the number of outputs to 26 (the size of the number of classes) and apply Softmax to generate a list of probabilities.

3.2.2 Functions

Each layer is activated using the Rectified Linear Unit function described in section 2.4. This means that neurons are activated if their input is greater than 0. Cross-entropy has been used to calculate loss, and the network is optimised using Adadelta. This has the effect of scaling learning based on the correctness of the output. Softmax has been used to enable one-hot encoding, whereby the largest value in the output array (normalised by Softmax) is set to one. All other probabilities are reduced to zero. The position of the 1 in the output array maps directly to the position in a dictionary of classes. For example, position 0 maps to A, position 1 maps to B and position 2 maps to C.

3.2.3 Visualisation of Convolutional Layers

Fig 3.5 shows the effect of the convolutional and pooling layers of Projnet when applied to the input image in Fig 3.4.

Figure 3.4: The letter A, taken from the EMNIST data set.

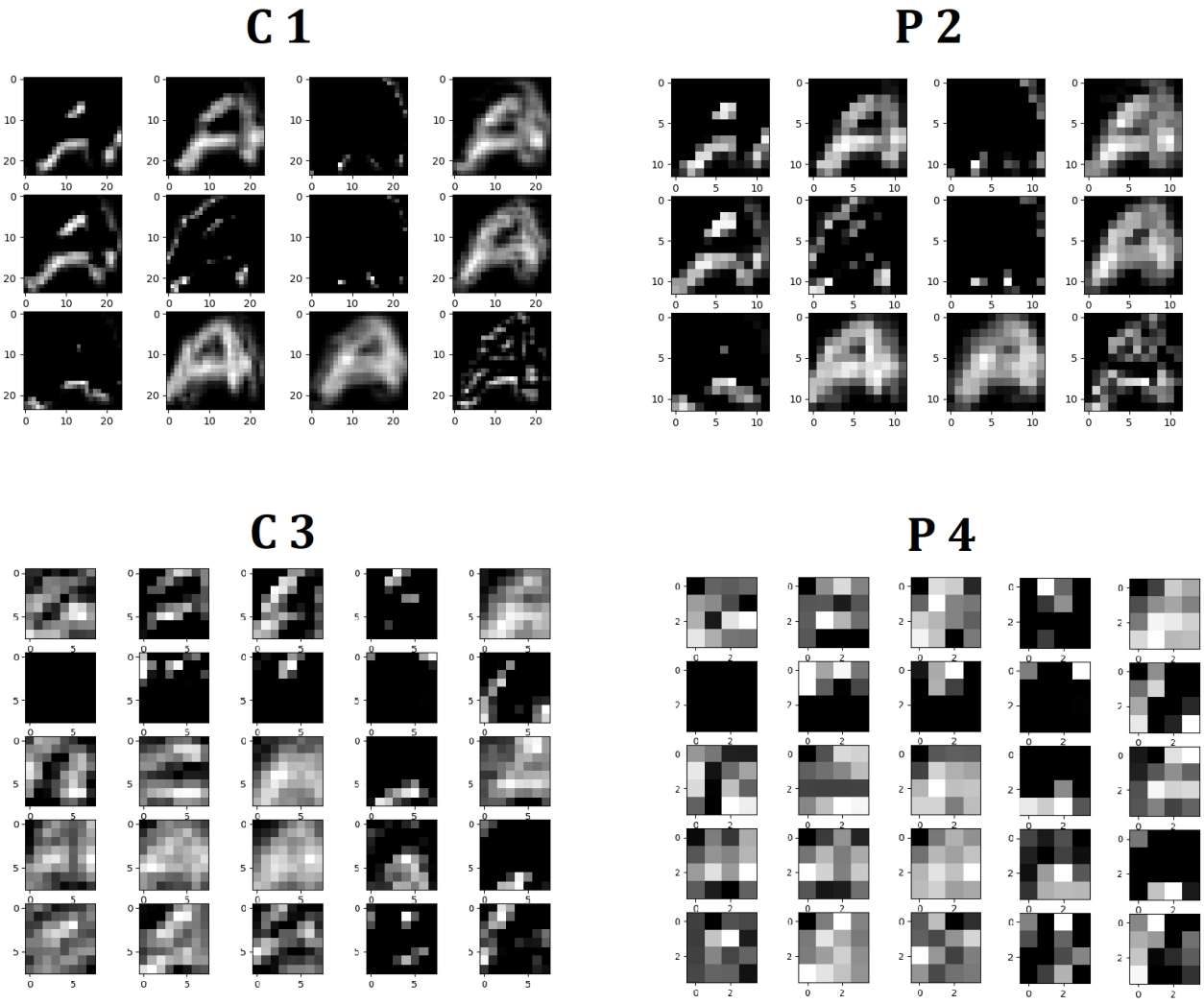


Figure 3.5: Showing the feature maps produced in each layer of the network.

3.3 Image Transformations

To increase the size and variety of the training data, various randomly generated transformations have been applied to individual data items at run-time. A transformation distorts an image, generating a variation of the original. Applying transformations to the entire data set creates new training data, which means that networks can be trained for longer without overfitting [Wong *et al.*, 2016]. As transformations are randomly applied at run-time, the generated images will be different on each epoch of training.

3.3.1 Rotate

The rotate transformation rotates each image randomly up to 20° clockwise or counter-clockwise.

3.3.2 Shift

The shift transformation randomly moves an image up to $0.2 \cdot img\_width$ on the x, y plane.

3.3.3 Zoom

Zooming increases or decreases the size of the features within an image, whilst mainting the images orignal size. Zooming can happen on the x or y axis, or both.

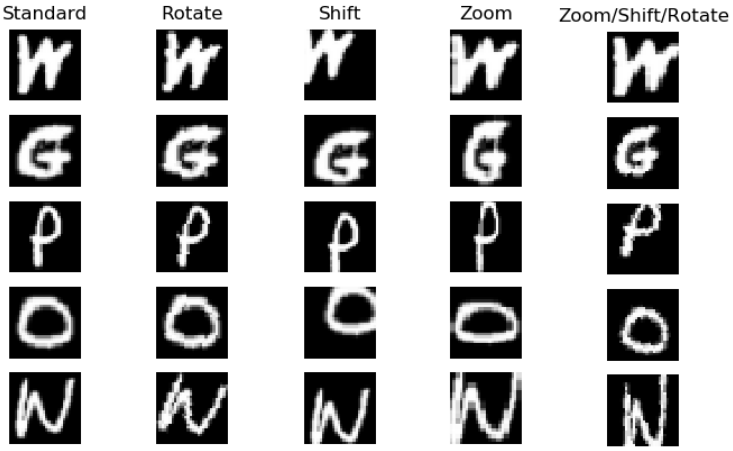


Figure 3.6: Training data, before and after transformations.

3.4 Ensemble Classifier

An ensemble classifier collates the outputs from multiple prediction methods. Current research suggests that ensemble methods can be used to make more accurate predictions than single classifiers [Opitz and Maclin, 1999]. Machine learning ensembles are comprised of a set of classifiers, the outputs of each individual classifier can be weighted to further increase the accuracy of the ensemble.

The ensemble classifiers used in this project give each of its composing classifiers outputs as probabilities. The sum of all outputs is divided by the number of classifiers to give an average prediction across all classifiers.

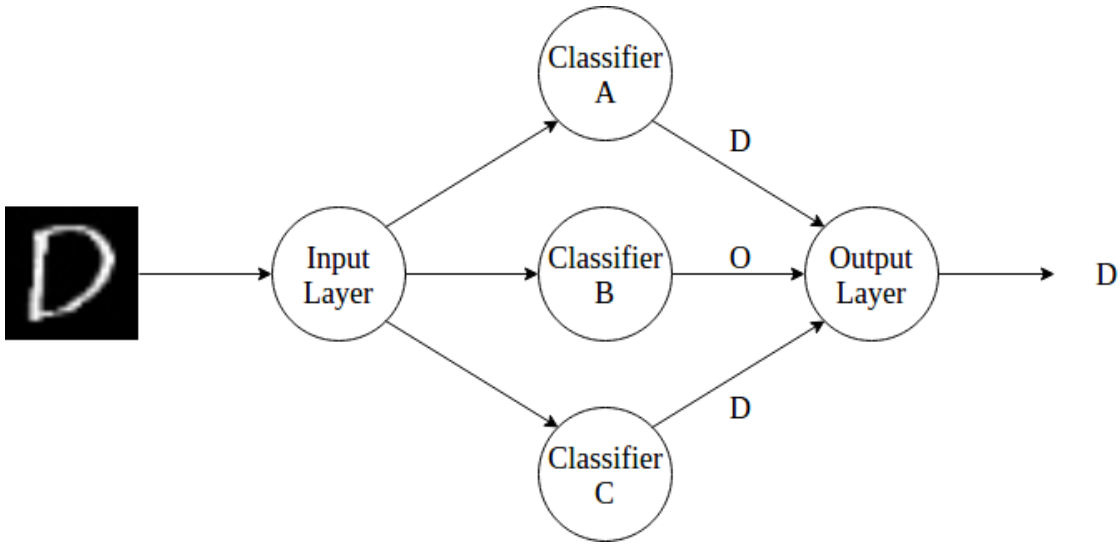


Figure 3.7: Visual representation of an ensemble classifier. The input (D) is classified by three separate classifiers, and the result is found by calculating their average prediction. In this example, classifiers A and C made the correct prediction (D), and classifier B was incorrect (O).

3.5 Classifier Training

To train a neural network on a data set, the data must be split into a training and testing set, usually at a ratio of 6:1. This is done so that during and after training, the performance of the network can be tested [Shiffman, 2012]. The data sets within EMNIST have already been split by providing separate files for the training and testing data. The training set is fed through the network x number of times, where the epochs trained is equal to x.

To ease the process of training networks using different parameters, a command-line program, *train_network.py* was created. It takes arguments to define the type of network, type of training data, the number of epochs to train on and the type of transformations to apply to training data. It generates a neural network using the Projnet architecture described in section 3.2, trains it using the specified data, and saves it to the disk. It also saves the training history, so that the performance of the classifier can be analysed.

The testing data set is used to validate the progress of the classifier after each epoch. Validation has no bearing on the direction of training as the backpropagation algorithm is not applied to the test data set. This is so that when we do a test at the end of training, the test data is unseen to the classifier, and therefore the results indicate how well the classifier can be applied to new data.

In training, there are four important metrics to measure:

1. Training Accuracy;
2. Training Loss;
3. Validation Accuracy;
4. Validation Loss.

3.5.1 Training Accuracy

The training accuracy is the accuracy of the network during training in the current epoch. This value is calculated by finding the percentage of correct classifications of the training data in the current epoch. As the classifier is recurrently trained using this data, it is only an indicator to its performance and it does not reflect the performance of the classifier when tested against new data.

3.5.2 Training Loss

The training loss is the value of the loss function calculated using the training data in the current epoch. Similarly to the training accuracy, the training loss should not be used to evaluate the performance of the classifier.

3.5.3 Validation Accuracy

The validation accuracy is the accuracy of the classifier when tested against the training data set. It is calculated at the end of each training epoch and again at the end of training. This is arguably the most important metric to measure as it shows how well the classifier performs against new data.

The percentage error can be calculated from the validation accuracy using the formula:

$$error = 100 - val\_acc \tag{3.1}$$

3.5.4 Validation Loss

The validation loss is the value of the loss function calculated on the testing data at the end of the current epoch. Generally, the validation loss is a good indicator as to the performance of the model. It is the value minimised by the optimisation function during the backpropagation algorithm.

3.5.5 Epochs

To ensure that the weights for a network are captured at the peak performing epoch of training, callbacks have been used to save the weights at the epoch with the lowest percentage error. Rather than saving a network at the end of training, each network is trained for set number of epochs. After each one, if the validation accuracy has increased from the epoch with the current highest validation accuracy, then the networks weights are saved. This training methodology ensures that the best weights are saved from a training run.

3.5.6 Model Selection

As specified in the Interim Planning and Investigation Report, the application was required to classify letters and numbers. To meet this requirement, three classifiers have been trained using the Projnet architecture specified in section 3.2. Each classifier was trained using a different data set depending on the classification task.

- To classify letters, the EMNIST-letters data set was used. This has been done to support a higher level of accuracy for documents containing only letters. To further improve the accuracy of this classifier, both upper and lower-case letters of the same type belong to the same class, for example p = P.
- To classify numbers, the EMNIST-digits data set was used. This has been chosen to support a higher level of accuracy for documents containing only digits.
- To classify both letters and numbers in the same document, the EMNIST-byclass data set was used. This is the largest data set with the widest range of classes.

An in depth analysis of the optimisation techniques explored to train these networks can be found in chapter 4.

3.6 Web Application Integration

The final part of this project involved the integration of the feature extraction algorithm and trained classifiers into an optical character recognition web application. A user can upload a photograph of a document or a piece of paper, and any text detected within the image will be converted to editable text. A mouse or touch pad can be used to draw onto a canvas, which allows for predictions to be made without uploading a photograph.

Users can log in to the application so that their classification history is saved, and can be downloaded as a text file at a later time. Users can also classify documents as a guest user, however this means that their classifications will not be saved.

The application has been deployed to a web server and can be found at:
"georgelancaster.pythonanywhere.com".

3.6.1 Architecture

The application can be split into three layers: Presentation, Business and Data. The Flask micro-framework was used to host and implement the back-end portion of the application. As this is a web application, the presentation layer was created using HTML. JavaScript, Jinja2 and AJAX were used for communicating between the presentation and business layers.

3.6.2 User Interface

The user interface is split into four web pages: the Login and Sign-up pages; the Hub page and the Prediction page. The Login and Sign-up pages are used for user authentication, a user can sign-up or sign-into the application. The Hub page shows the history of all classifications for the authenticated user and the Prediction page allows for images to be classified.

As this project did not focus on usability and design, the user interface was kept minimalistic. Big buttons and a consistent colour scheme ensure that it is simple and intuitive to use. This also meant that there is a minimal reliance on external JavaScript libraries. The two libraries used are: Cropper, for cropping images on the predict page; and jQuery for the use of AJAX.

3.6.3 Flask

Flask was chosen as the web framework because it is highly customisable and can easily be used with Keras as it is written in Python. Flask operates at a high-level, this is beneficial as it meant that minimal development time was spent on the underlying architecture of the application.

Flask uses HTTP operations as annotations to functions in Python code to communicate between the client and server. Not only is this elegant in design, it is easy to route a button to a function in the server-side code.

3.6.4 Database

The database has been created using SQLAlchemy, which is a database toolkit for Python. It stores user login information and prediction history. The user table holds the user ID, username and password for a user. All user passwords have been hashed. In the case of security breach, the passwords are encrypted. The prediction table holds a prediction ID, image reference and a prediction result. It also has a user ID foreign key to link a prediction to a user. This enables the hub page to show all predictions for a user.

The image reference is stored as a string and links to a location in the project folder, using the name of the user as the files containing directory. Image predictions are resized to save space on the server. The result of a prediction is saved as a string, new lines are denoted with the regular expression "\n".

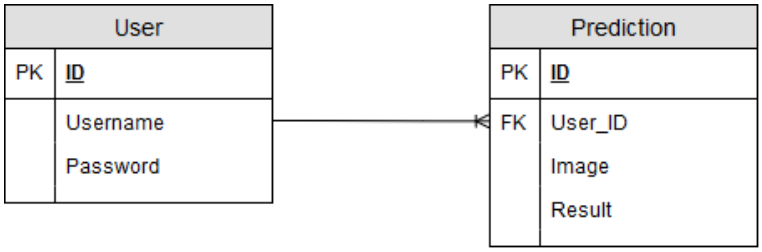


Figure 3.8: Entity relationship diagram, which shows the relationship between tables in the database.

3.6.5 Deployment

The application has been deployed using a two-tier client-server model. The database is hosted on the same server as the Flask application. Although this reduces the modularity of the system, it decreases the time it would take to connect to an external database using TCP.

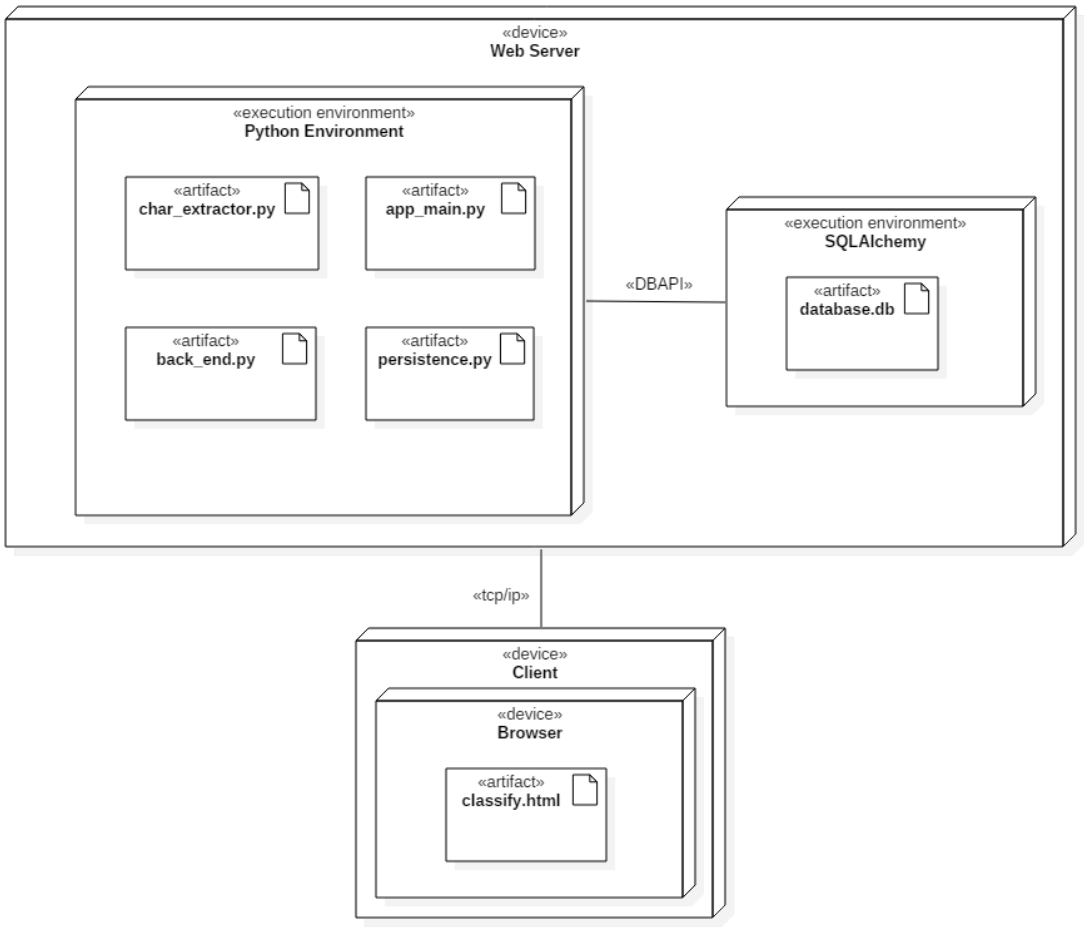


Figure 3.9: Deployment diagram, which shows how the application has been deployed to PythonAnywhere.

Chapter 4

Classifier Evaluation

Four experiments were done to find the best parameters for training Projnet. This was done using the command-line program *test_network.py*, which tests a classifier against a data set, creating two graphics to evaluate its performance. The first graphic is a confusion matrix, showing the predicted labels against the actual labels. The second is a graph showing how the training accuracy and loss, and validation accuracy and loss change throughout training. To increase the accuracy of the classifiers, upper and lowercase letters of the same type have been merged into a single class; the letter 'r' will be classified the same as the letter 'R'. All experiments have been performed on the Projnet classifier architecture described in chapter 3.3.

4.1 Experiment 1 - Evaluate the effect of Training Time on Performance

4.1.1 Aim

The aim of this experiment is to find the optimum number of epochs to train the Projnet classifier architecture described in chapter 3.3.

4.1.2 Methodology

Six identical classifiers have been trained for two hundred epochs using the EMNIST-letters data set. Callbacks have been used to log the training history, and to save the classifier at the epoch with the lowest percentage error. Training is inherently stochastic as the weights of each classifier are randomly initialised, and the order in which the training data is fed to the classifier is random. Each classifier will learn at a different rate and peak at a different epoch.

There are 20,800 letters in the test data set.

4.1.3 Results

Classifier Number	Saved Epoch	Incorrect Classifications	Percentage Error	Test Loss
1.1	160	1201	5.77	0.24
1.2	74	1204	5.79	0.19
1.3	90	1227	5.90	0.21
1.4	127	1204	5.81	0.21
1.5	129	1188	5.71	0.20
1.6	112	1197	5.75	0.20

Table 4.1: Table of results from experiment 1, summarising the accuracy of various classifiers trained using the EMNIST-letters data set.

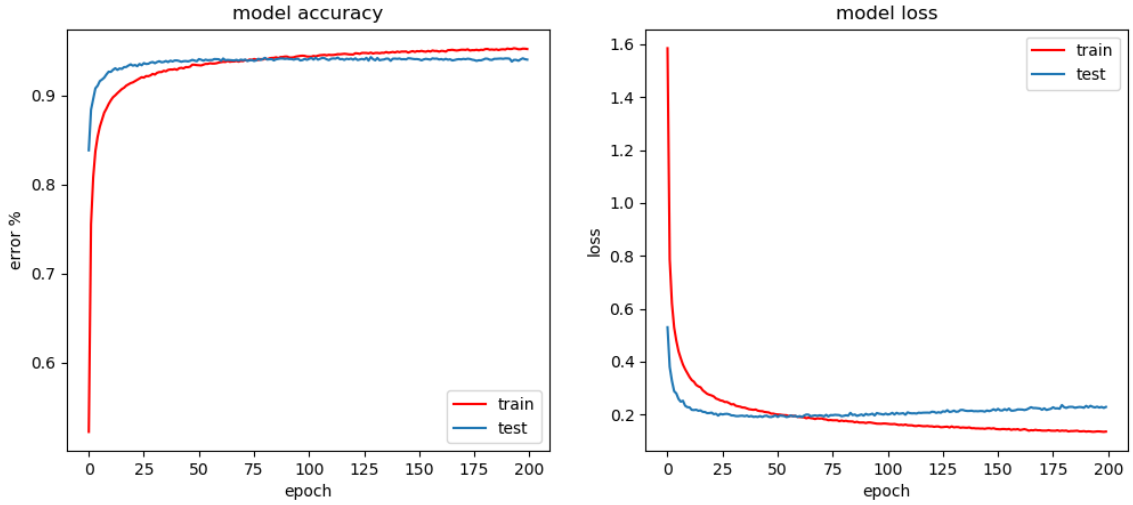


Figure 4.1: Training graphs of classifier 1.5, showing changes in model accuracy and loss against number of epochs during training.

Although the best performing classifier peaked at epoch 129, Fig 4.1 suggests that overfitting starts to occur at epoch 65 when the test loss stagnates, and the training loss intersects with the test loss. The training accuracy overtakes the test accuracy at approximately epoch 75, further indicating that this is when the model starts to overfit.

The wide range of saved epochs do not give a definitive, optimum epoch for training. However, we can assume that since no classifiers were saved at a number of epochs greater than 160, or less than 74, that any value between these two can be considered optimal.

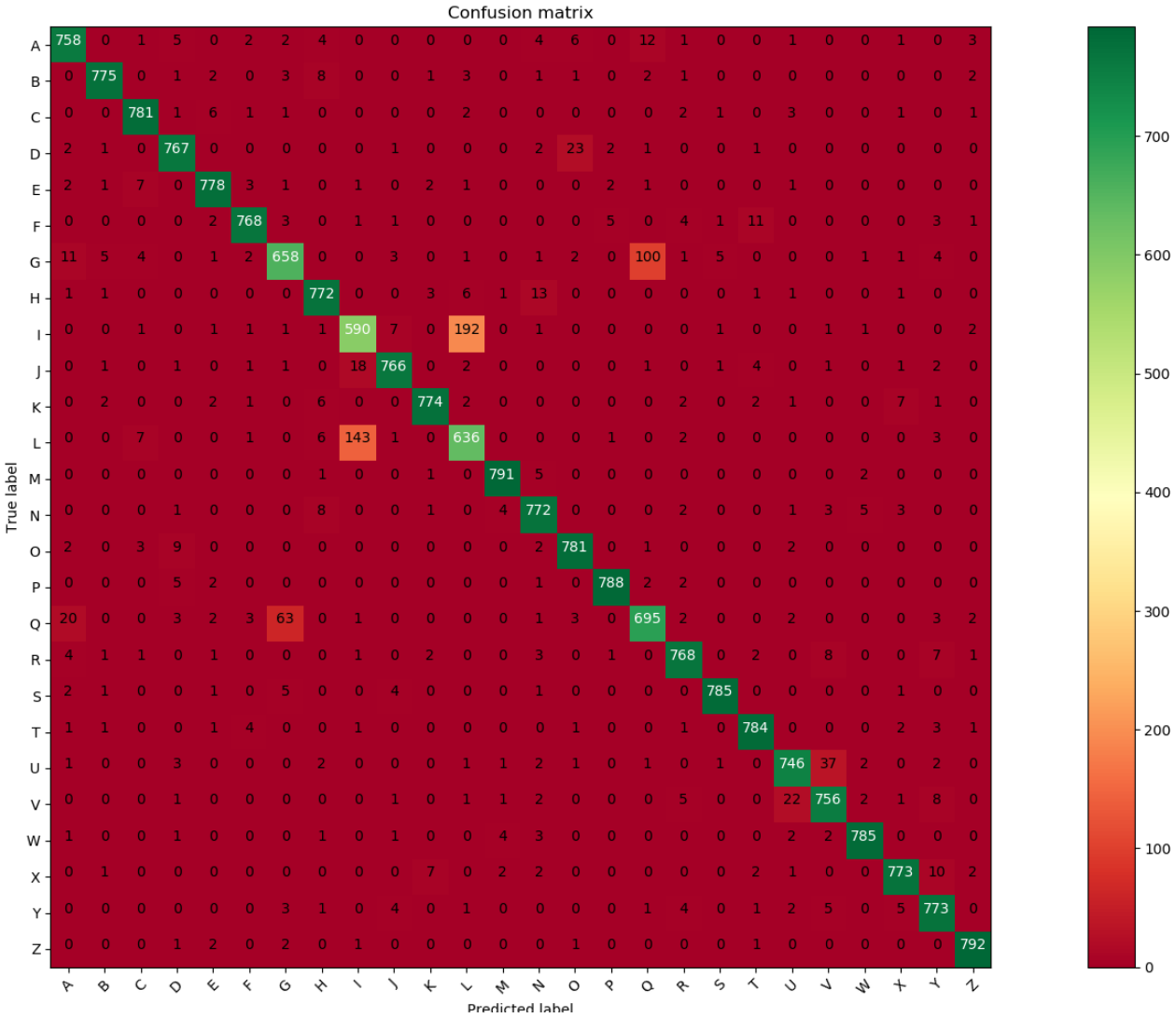


Figure 4.2: Confusion matrix for classifier 1.5, showing the number of predictions for each data item plotted against its true value.

The most misclassified case is ‘I’ against ‘L’, followed by ‘L’ against ‘I’. ‘L’ is being mistaken for ‘I’ 24 per cent of the time and ‘L’ is mistaken for ‘I’ 17.875 per cent of the time. This may be due to these characters being represented similarly throughout the data set, causing the classifier to learn similar neuron activations for both classes.

4.2 Experiment 2 - Evaluate the Performance of Networks Trained using Transformed Data

4.2.1 Aim

As referenced in section 3.3, image transformations can be applied to increase the size and variety of the training data. In theory, a model can be trained for longer without overfitting. The slight variations in the data caused by transformations prevents the classifier from learning the data set directly, thus it learns the characteristics of each class. It is important to only apply subtle changes to the data, as large modifications can make data items unrecognisable.

The aim of this experiment is to see if applying image transformations to the training data improves the accuracy of the classifier when tested against the EMNIST-letters test dataset.

4.2.2 Methodology

Seven classifiers have been trained for 300 epochs, using different combinations of image transformation. The networks in this experiment have been trained for longer than in experiment 1, to measure the effect transformations have on overfitting. Callbacks have been used to save the classifier at the epoch with the highest validation accuracy.

Transformation	Value
Rotate	Random rotation range 0-15°
Shift	Random shift on x, y plane $(0.2 \cdot area)$
Zoom	Random zoom in range 0.3

Table 4.2: Table showing the exact values of the applied transformations.

4.2.3 Results

Classifier Number	Transformations	Saved Epoch	Incorrect Classifications	Percentage Error	Test Loss
2.1	Zoom	210	1138	5.47	0.17
2.2	Rotate	300	1123	5.40	0.18
2.3	Shift	244	1291	6.21	0.19
2.4	Zoom/Rotate	244	1126	5.41	0.17
2.5	Zoom/Shift	253	1403	6.75	0.20
2.6	Shift/Rotate	251	1384	6.65	0.20
2.7	Shift/Rotate/Zoom	267	1447	7.00	0.21

Table 4.3: Table of results from experiment 2, summarising the accuracy of various classifiers with image transformations applied to their training data.

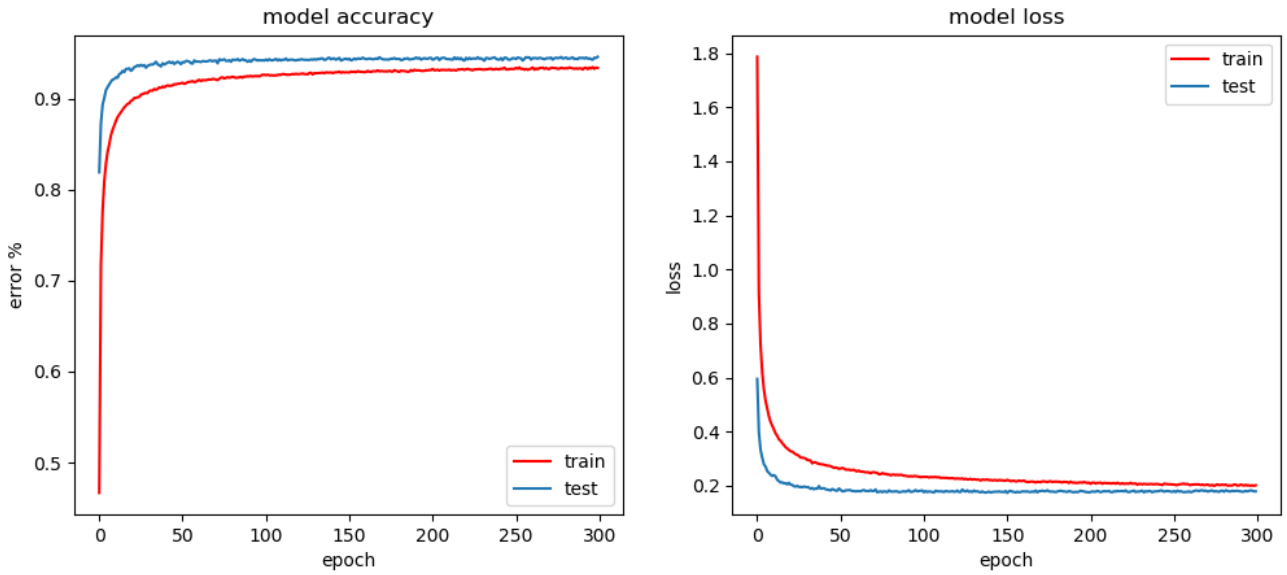


Figure 4.3: Training graphs of classifier 2.2, showing changes in model accuracy and loss against number of epochs during training.

As expected, some mild transformations to the training data improved the accuracy of the classifer when tested against the test data set. Using the shift transformation in any of the classifiers always caused a lower accuracy than using no transformation method. Decrease in accuracy with the shift transformation may be as a result of important features being moved outside of the scope of the image. Another reason for this could be that all letters within EMNIST are centered, and moving them off center could cause neuron activations for other classes.

The zoom and rotate transformations provided the best results when used alone or together. Classifiers 2.1, 2.2 and 2.4 outperformed classifier 1.5 by achieving a lower percentage error. Fig 4.3 suggests that overfitting was about to occur when the training ended at epoch 300,

as the training and testing loss were about to intersect. All of the classifiers trained using transformations peaked much later than any of the classifiers trained using the unmodified data set. This suggests that the experiment was successful, and that transformations to images can be used to increase the size and variety of the training data, which leads to greater training times before overfitting occurs.

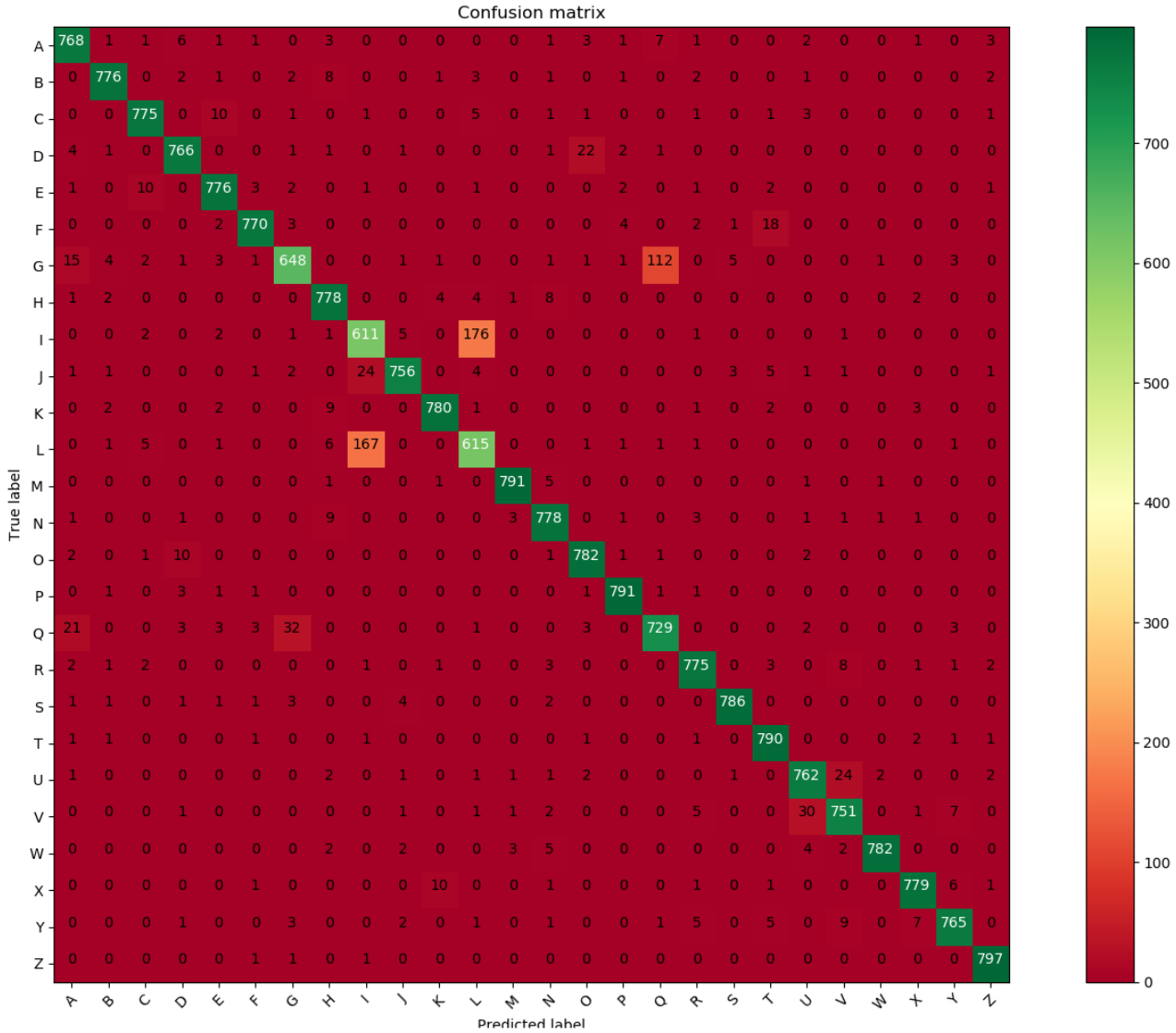


Figure 4.4: Confusion matrix for classifier 2.2, showing the number of predictions for each data item plotted against its true value.

Fig 4.4 shows that the most common misclassifiactions were between the letters 'L' and 'I'. There is a similar distribution for the number of incorrect predictions for each class in classifiers 1.5 and 2.2.

4.3 Experiment 3 - Evaluate the Performance of Ensemble Classifiers

4.3.1 Aim

Ensemble classifiers have been successfully used in other studies as a method of improving classification accuracy. This experiment aims to analyse how ensembles compare to lone classifiers in the task of recognising hand-written letters.

4.3.2 Methodology

Three ensemble classifiers have been created using multiple classifiers from experiments 1 and 2, and tested against the EMNIST-letters dataset. Classifier 3.1 uses the classifiers from experiment 1, classifier 3.2 uses the classifiers from experiment 2, and classifier 3 uses only classifiers with a percentage error of less than 5.80.

4.3.3 Results

Classifier Number	Combined Classifiers	Incorrect Classifications	Percentage Error	Test Loss
3.1	1.1, 1.2, 1.3, 1.4, 1.5, 1.6	1083	5.21	0.16
3.2	2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7	1084	5.21	0.16
3.3	1.1, 1.2, 1.5, 1.6, 2.1, 2.1, 2.4	1037	4.99	0.15

Table 4.4: Table of results from experiment 3, summarising the accuracy of various ensemble classifiers.

Table 4.4 shows that the best performing classifier was 3.3, achieving the lowest percentage error of 4.99. This is the lowest error from any of the three experiments and suggests that ensembles can be used to create more accurate classifiers. The ensemble has a lower percentage error than any of its composing classifiers as a result of the synergistic effect of collating outputs.

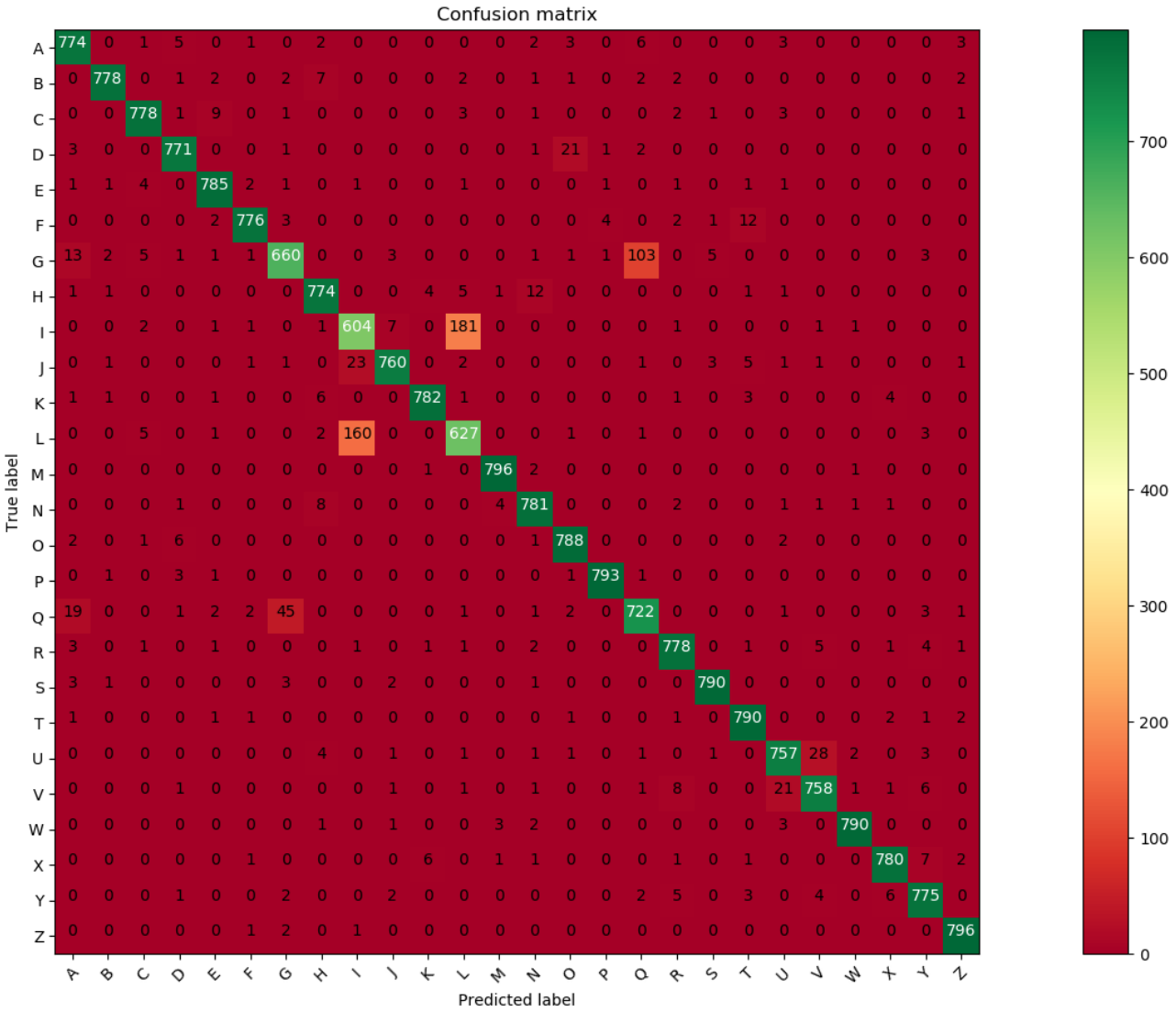


Figure 4.5: Confusion matrix for network 3.3, showing the number of predictions for each data item plotted against its true value.

Fig 4.5 follows a similar trend as observed in Fig 4.2 and Fig 4.4, showing that the most misclassified classes are 'I' and 'L'. This supports the assumption that there may be ambiguities in the data.

4.4 Experiment 4 - Evaluate the Performance of Classifiers on a New Data Set

4.4.1 Aim

This experiment aims to analyse the performance of all previously tested classifiers on the PDS data set, which was created using the *char_extractor.py* module for this project. This has been done to estimate the accuracy of the web application. It is also an indication as to how well the classifiers can be generalised to solve real-world image recognition tasks.

4.4.2 Methodology

Each of the classifiers defined in experiments 1, 2 and 3 have been tested against the PDS data set. Each classifier was used to predict all data items within PDS. The testing was carried out using the *test_network.py* module.

4.4.3 Results

Classifier Number	Percentage Error	Test Loss
1.1	6.61	0.40
1.2	6.47	0.28
1.3	6.87	0.30
1.4	7.13	0.34
1.5	6.21	0.31
1.6	5.68	0.30
2.1	4.62	0.20
2.2	4.89	0.22
2.3	9.64	0.36
2.4	3.30	0.17
2.5	9.91	0.35
2.6	5.81	0.30
2.7	7.40	0.27
3.1	5.42	0.21
3.2	3.83	0.20
3.3	3.70	0.18

Table 4.5: Table of results from experiment 4, summarising the accuracy of all classifiers when tested against the Project Data Set.

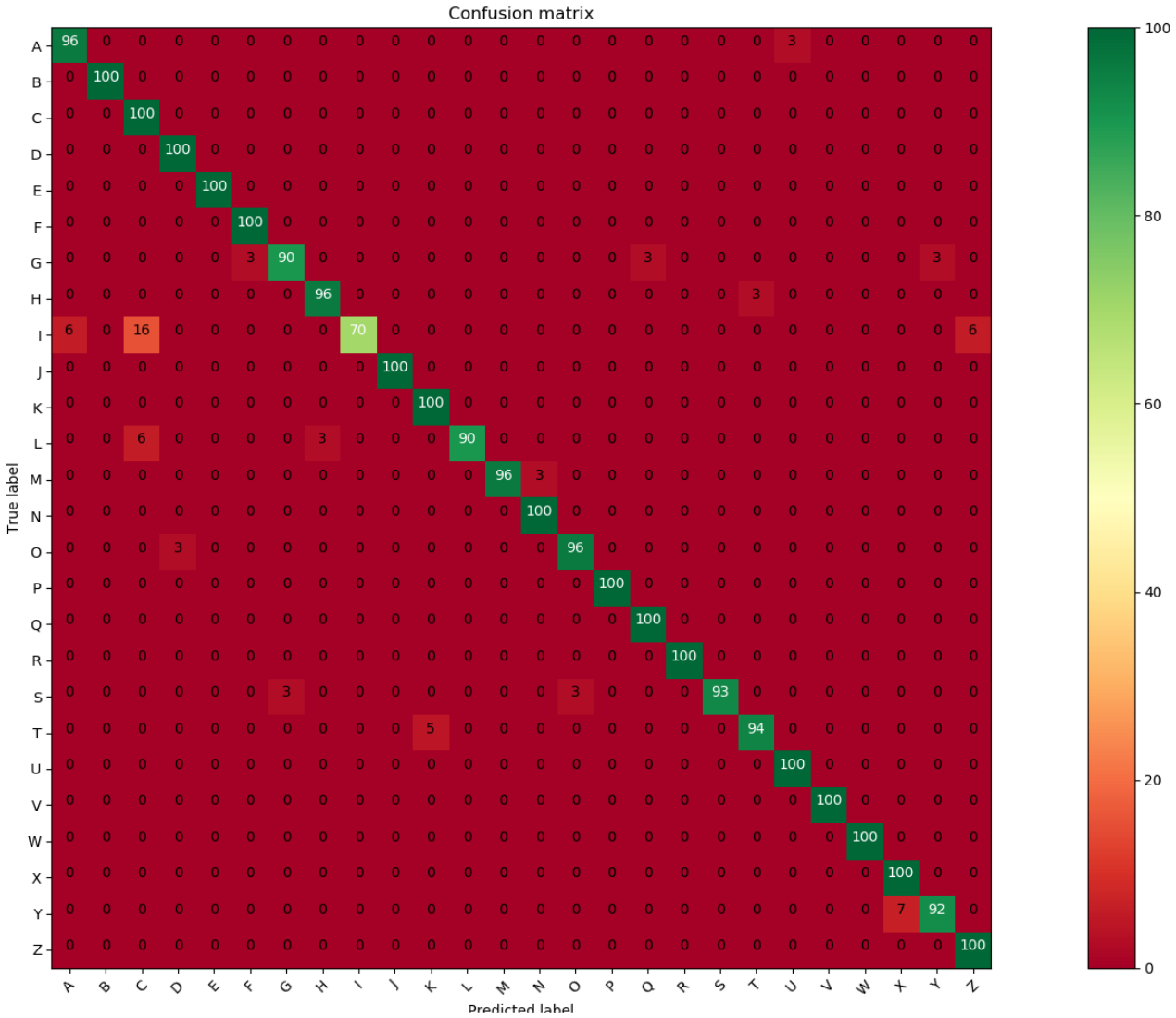


Figure 4.6: Confusion matrix for network 2.4 when tested on PDS. The data has been normalised to values between 0 and 100 as PDS is not balanced.

Classifier 2.4 achieved the best results from this experiment, with a percentage error of 3.30, exceeding its performance when tested against the EMNIST-letters test data set by 2.11 per cent. The second best performing classifier, 3.3, achieved a 3.70 percentage error, which is lower than its score against the EMNIST-letters data set by 1.29 per cent. The results suggest that the classifiers can be applied to classify new data. The percentage error from these classifiers should represent an estimate as to the accuracy of the web application.

Fig 4.6 differs from the previous trend, in that 'I' and 'L' are no longer the most falsely predicted classes. This could be due to a greater variance in the way in which these letters have been represented in PDS.

4.5 Digit Classification

Training Projnet on the EMNIST-digits data set using the rotate and zoom transformations gave a loss of 0.013 and a percentage error of 0.29. The results are better than when trained using the EMNIST-letters dataset as there are only ten classes to classify. Out of 40,000 test predictions, only 116 were incorrectly predicted.

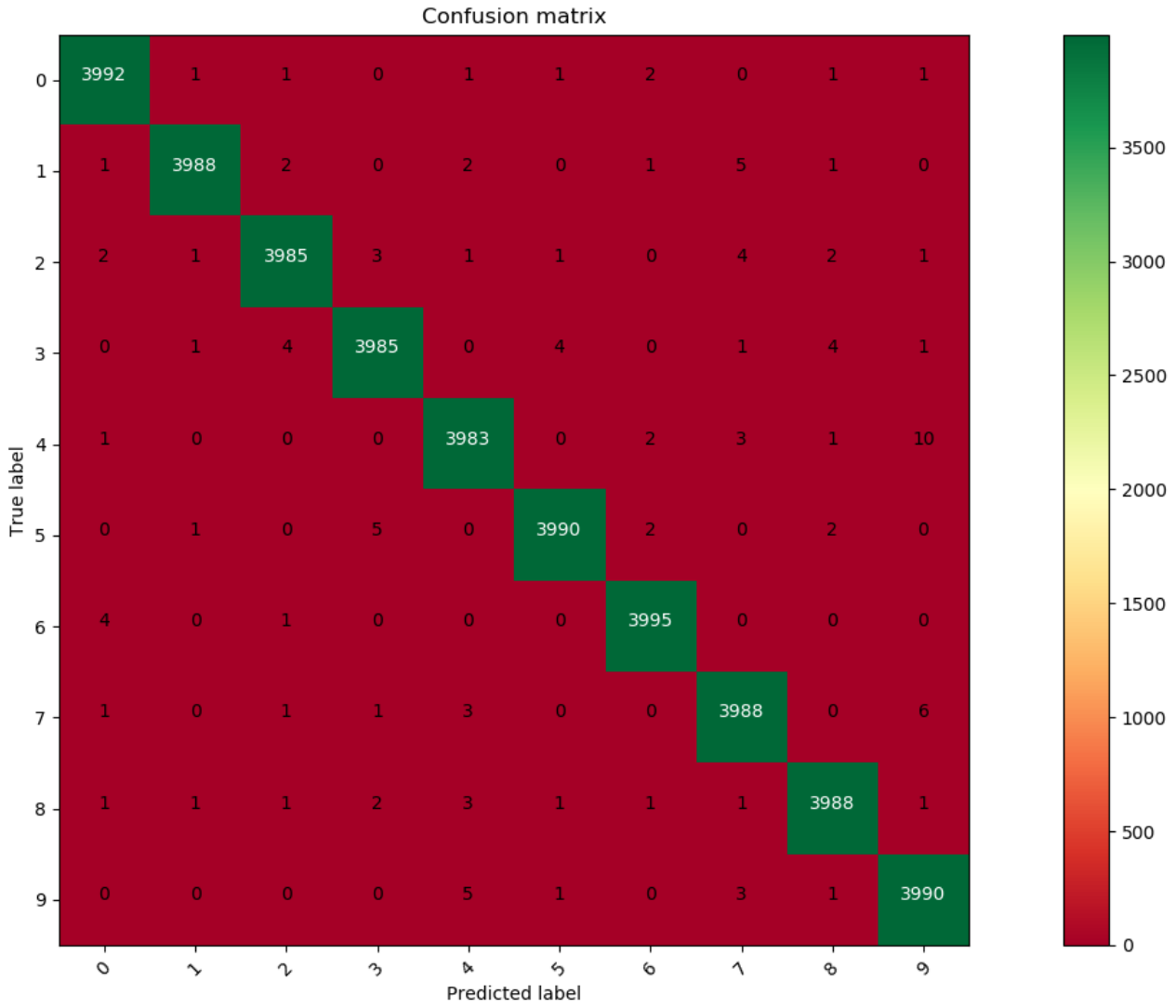


Figure 4.7: Confusion matrix for Projnet when trained and tested using the EMNIST-digits data set. There are 4000 digits in each class.

4.6 Universal Character Classification

Training Projnet on the EMNIST-byClass dataset yields a universal character classifier. It can classify uppercase and lowercase letters, as well as numbers. It validation loss is 0.39, and its percentage error is 14.15.

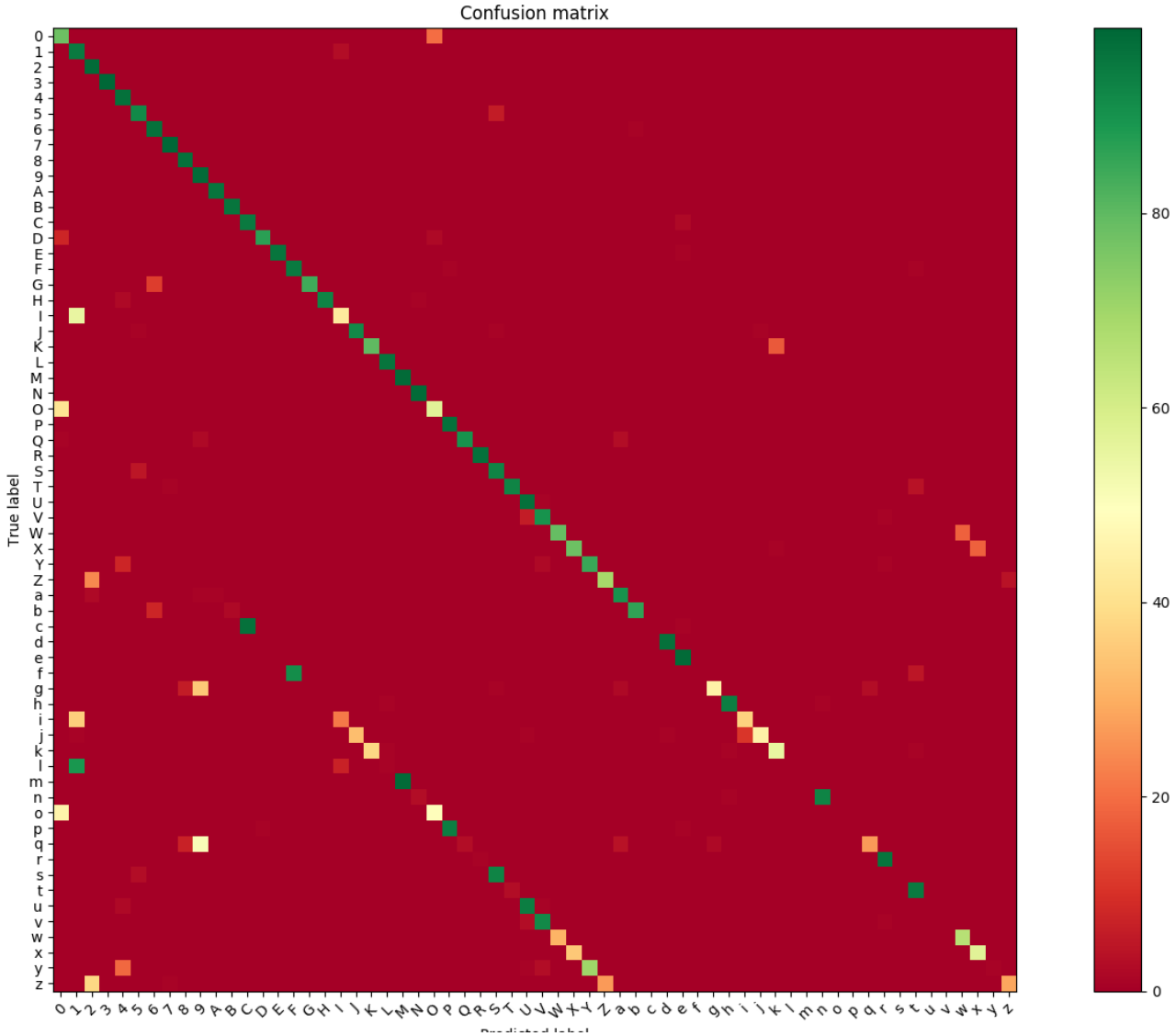


Figure 4.8: Normalised confusion matrix for Projnet when trained and tested using the EMNIST-byclass data set.

There are two reasons why the accuracy is lower than in the other classifiers tested. The first being that there are more classes to fit data into. The second is that many uppercase and lowercase letters look the same. Many lowercase letters are mistaken for their uppercase counterparts, as shown in Fig 4.8.

4.7 Discussion

Classifier 3.3 has been used in the web application to classify letters as it has the lowest mean percentage error of 4.45 when tested against the EMNIST-letters and PDS data sets. Classifier 2.4 was the second most accurate classifier, with a mean percentage error of 4.36, which is 0.01 per cent less than classifier 3.3. This suggests that applying transformations to training data, and collating successful pre-trained classifiers as ensembles can generate better results than using traditional trainining methods.

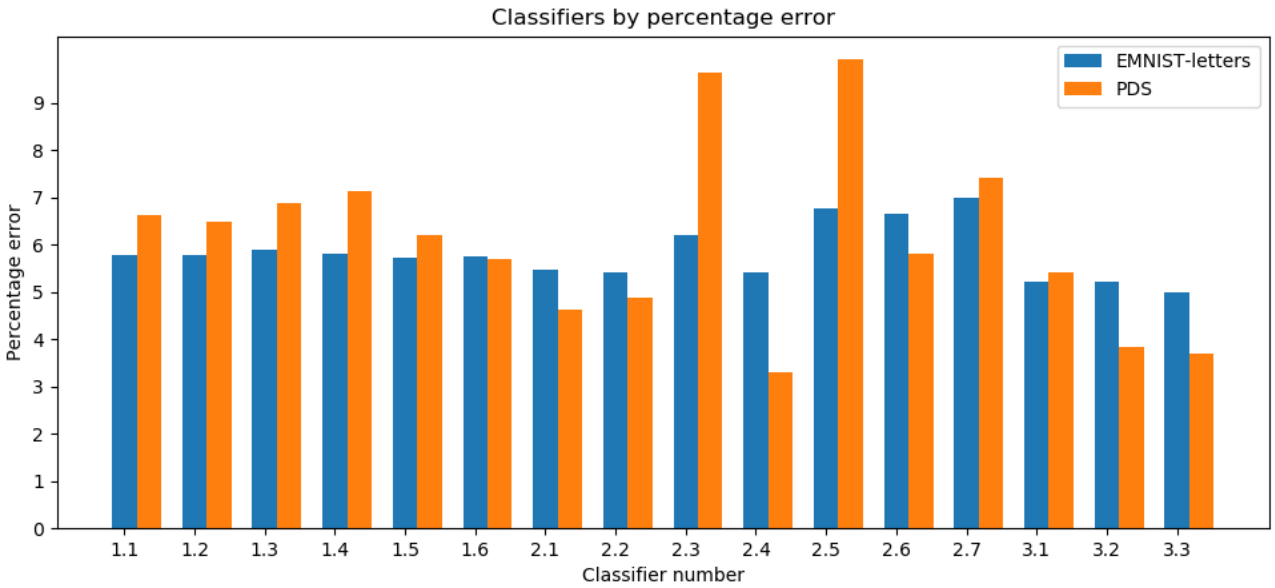


Figure 4.9: Bar chart showing the percentage error of each classifier when tested the EMNIST-letters and PDS data sets.

Due to the time it would take to train three ensemble classifiers on the EMIST-digits and EMNIST-byclass data sets, the same techniques used to train network 2.4 have been applied to train the digit and universal classifiers, which have been used in the web application. This involves training for 300 epochs using the zoom and rotate transformations, and saving the weights at the epoch with the lowest percentage error.

The current record for classifying the original MNIST data set [LeCun, 1998] is 0.23, by a network that uses roughly as many sparsely connected layers as found in mammals between retina and visual cortex [Ciregan *et al.*, 2012]. Projnet achieved a percentage error of 0.29 using just seven layers, but trained on the EMNIST-digits dataset, which contains four times as much data.

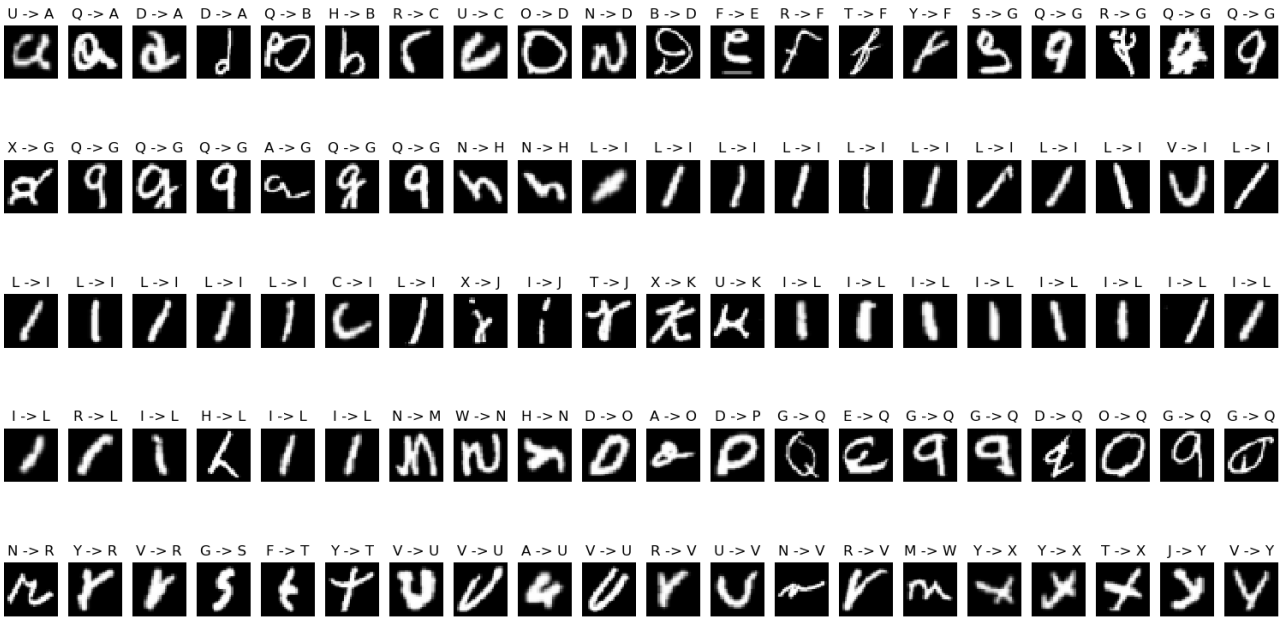


Figure 4.10: Shows 100 misclassifications from testing classifier 1.5 against EMNIST. The captions are in the form prediction -> label.

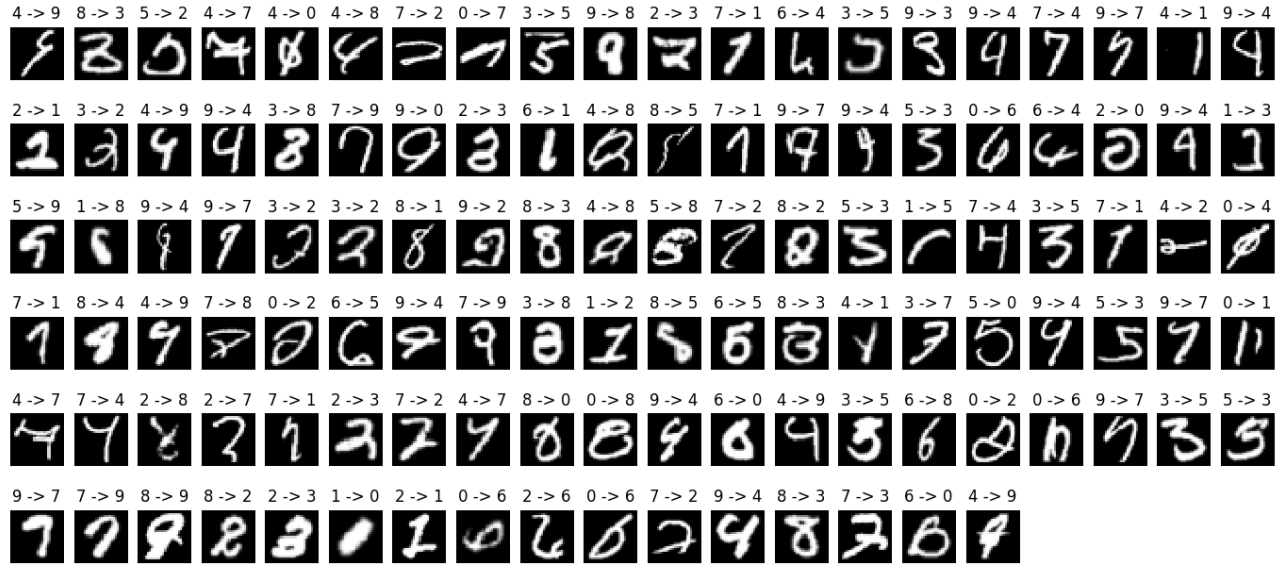


Figure 4.11: Shows the 116 misclassifications from testing Projnet against the EMNIST-digits data set. The captions are in the form prediction -> label.

Fig 4.10 and 4.11 show that although some of the misclassifications may be easy to determine for humans, there are many genuinely ambiguous cases, which may have influenced the results of these experiments. Badly drawn examples of a letter could cause the model to unlearn previously learned correct features.

Chapter 5

Project Management and Planning

The constant research being done throughout this project made it difficult to plan ahead. As new knowledge was gained, parts of the implementation required re-writing to use more effective methods. Whilst this improved the outcome of the project, it made it difficult to plan ahead as it was impossible to predict what new techniques would be discovered. This made the project suitable for the agile methodology. Work was completed in small sprints after research had been completed on specific topics.

In the planning stage, a table of milestones and deliverables was created. It was important to have a working prototype completed by December 15th so that the rest of development time could be spent on optimisation and experimentation.

Goal Date	Completion Date	Milestone
14/11/17	10/11/17	Have a prototype ready in Python
22/11/17	22/11/17	Hand in planning and investigation report.
28/11/17	28/11/17	Viva voce with project supervisor and second reader.
01/01/18	15/12/17	Completion of prototype
01/01/18	01/01/18	Analyse initial results
01/01/18	01/01/18	Continue with improvements
09/03/18	15/02/18	Begin writing report for examiners
25/04/18	25/04/18	Complete folio page
09/05/18	08/05/18	Hand in deliverable

Table 5.1: Table showing the predicted date of completion next to the actual completion date.

5.1 Diary

A diary was kept from the beginning of this project and can be found in appendix C. Although the diary gives a good indication as to the progression throughout the project, it is not complete and does not show every day which was worked on the project.

Chapter 6

Reflection

6.1 Background Research

It would be impossible to learn the entirety of machine learning in the short space of time given to complete this project, therefore it was important to focus research on studies and projects similar to this one. The research of Yann LeCunn was especially useful, as the neural network Lenet-5 was the inspiration for the Projnet classifier used in this project [LeCun *et al.*, 1998a].

6.2 Learning Points

6.2.1 Technical Skills

Having had very limited prior experience of machine learning, a lot of time was spent learning the theory behind neural networks. It would have been possible to complete this project without knowing how Keras and TensorFlow works, but it was important for the projects success to understand their underlying technology to use them effectively.

Python was learned solely for the purpose of completing this project after it was determined to be the most appropriate language. This meant learning how to write Python in a short amount of time.

It was important to develop web development skills to create the web-application user interface. More importantly, learning to using Flask and HTTP operations allowed for communication between the client and server. Although only a small part of the project, JavaScript was used on all of the web pages, so it was important to learn how to use it effectively.

6.2.2 Project Management

The experience of working on this project has been valuable in the development of time-management skills, as well as learning the importance of documenting progress throughout a project. Given the time that was available to complete this project whilst also being required to complete other assignments, it was important to divide time equally amongst tasks and to prioritise the most important features.

The work diary was by far the most important project management technique used. Not only did it allow for progress to be tracked, but it also allowed for the planning of future work.

6.3 Areas of Improvement

6.3.1 Increase Accuracy

As discussed in the introduction to this report, the classification of hand-written characters has never been achieved with 100 per cent accuracy. This leaves the project open to improvement by testing deeper network configurations than Projnet, which have been used in other studies to achieve better results [Ciregan *et al.*, 2012].

Accuracy could also have been improved by extending PDS to include a training data set. This would have been used as an extension to EMNIST to allow for training to be done on more examples. Implementing this would have required better organisation, and maybe participation from the university to find more participants to gather data.

Two image transformation techniques that have had the most success in other studies are affine transformation and elastic distortion [Lauer *et al.*, 2007] [Simard *et al.*, 2003]. It is predicted that even better results could have been achieved if these techniques had been used to expand the EMNIST data set.

6.3.2 Word Formatting

Given more time to work on this project, it would be spent investigating other techniques for formatting classification results. Currently, line formatting is preserved on output, but the spaces between words are not. This could be implemented using a similar algorithm to the one used for line spacing, but applying it to the x-axis, splitting a line into words.

6.3.3 Punctuation

This project could be extended through the detection of punctuation. It was not possible to do this as there were no easily accessible data sets that include punctuation, which were suitable for this project. Given the time contraints of this project, creating a new data set was not an option. Punctuation also brings a greater challenge due to its size compared to characters.

Currently, the feature detection algorithm removes features with an area less than 200 pixels, therefore the current implementation would need to be changed.

6.3.4 Languages

This implementation only works for letters found in the English alphabet. This is because of the difficulty in finding a data set that includes other alphabets, as well as the tim]e it takes to train a single classifier. Using a setup with an Nvidia Geforce 1050ti, training using the EMNIST-letters data set with image transformations takes approximately 35 seconds per epoch. Resulting in a training time of up to three hours. Adding more classes only increases the time it takes to train a network, and decreases the accuracy of the network once trained as there are more features to learn. Languages which use other alphabets often use letters with accents, further making it more difficult to distinguish between classes.

6.3.5 Data Cleansing

As shown in chapter 4.7, some of the data items within EMNIST are poor representations of their class. Further research could be done to test the effect on classifier accuracy by removing these data items.

6.3.6 Standard of Code

Having had more experience using heavily object-oriented languages like Java, it took some time to adjust to writing code in a more 'Pythonic' way. Code written in Python aims to be clear, concise and maintainable.

The guidelines laid out in the PEP-8 document have been followed to conform to the relevant standards for writing Python code where possible [van Rossum *et al.*, 2001]. A lot of development time was spent learning how to write effective Python. Because of this, some of the code written in early development had to be re-written to meet these standards.

There is a minimal amount of JavaScript and, as a result of this, not much time was spent on learning how to write it to a high standard. The standard to which it was written could be improved, however it performs its functionality as intended.

6.4 Technical Challenges

For a long time, data items from PDS were being predicted with poor results. As the EMNIST data set has been normalised, a classifier trained using EMNIST will expect to predict normalised images. This was solved by normalising the pixel values within the PDS images to values between 0 and 1, as has been done with EMNIST.

The feature extraction algorithm described in section 3.1 proved to be one of the greatest challenges in this project. It took multiple failed implementations before finding one that worked. Originally, the algorithm was going to split an image into paragraphs, words then letters. This was not possible due to the varying sizes of text found within documents in testing. This meant that the approach had to be changed to detect single characters, then order them by their position in the image.

6.5 Experimentation Results

The results from this project indicate that it has been highly successful. In the interim planning and investigation report, a requirement was to classify hand-written upper-case letters to at least 70 per cent accuracy (30 per cent error) for the project to be considered a success. The highest accuracy achieved by any of the networks, when tested against PDS was a 3.30 percentage error, surpassing this goal by 26.70 per cent. Similarly, the goal for classifying hand-written digits was 70 per cent accuracy (30 per cent error). A result of 0.29 percentage error was achieved, beating this goal by 29.71 per cent.

Classifier 3.3 achieved a percentage error of 4.99 per cent when tested against the EMNIST-letters dataset. When applying it to PDS, there was an increase in performance. This could be in part due to the smaller sample size of data, and that it was only generated by two people. This meant that there was less variance in the way the letters were drawn.

6.6 Conclusion

From a project management viewpoint, this project could be considered to be successful. The project was finished on time, and many of the goals were completed before their deadlines. The agile methodology suited this project well, and was flexible enough to allow for design decisions to me made mid-development.

Neural networks are extrememly powerful tools and give impressive results when applied to a specific task like character recognition. This project has shown that even simple networks can give excellent results, and that networks that perform well in testing can be applied to real-world tasks.

Many people treat neural networks as black-box systems, which generate accurate predictions without the developer having any knowledge of their internal workings. It was important for the success of the project to not use neural networks in this way, and to use an understanding of how they work in optimisation to achieve better results.

Bibliography

[Abadi *et al.*, 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., *et al.* (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16 (pp. 265–283).

[Boyle and Thomas, 1988] Boyle, R. D. and Thomas, R. C. (1988). *Computer vision: A first course*. Blackwell Scientific Publications, Ltd.

[Bradski and Kaehler, 2000] Bradski, G. and Kaehler, A. (2000). Opencv. *Dr. Dobb's journal of software tools*, 3.

[Canny, 1987] Canny, J. (1987). A computational approach to edge detection. In *Readings in Computer Vision* (pp. 184–203). Elsevier.

[Carling, 1992] Carling, A. (1992). *Introducing neural networks*. Sigma Press.

[Chollet, 2018] Chollet, F. (2018). keras. [Online; accessed 10 April 2018] https://github.com/fchollet/keras.

[Christopher, 2016] Christopher, M. B. (2016). *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York.

[Ciregan *et al.*, 2012] Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on* (pp. 3642–3649).: IEEE.

[Cohen *et al.*, 2017] Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*.

[Cornell, 2018] Cornell (2018). Neural networks and machine learning. [Online; accessed 01 May 2018] http://blogs.cornell.edu/info2040/2015/09/08/neural-networks-and-machine-learning/.

[Diem *et al.*, 2013] Diem, M., Fiel, S., Garz, A., Keglevic, M., Kleber, F., and Sablatnig, R. (2013). Icdar 2013 competition on handwritten digit recognition (hdrc 2013). In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on* (pp. 1422–1427).: IEEE.

[Glorot *et al.*, 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315–323).

[Goodfellow *et al.*, 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. [Online; accessed 01 May 2018] http://www.deeplearningbook.org.

[Haykin, 2004] Haykin, S. (2004). A comprehensive foundation. *Neural networks*, 2(2004), 41.

[Hubel and Wiesel, 1968] Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1), 215–243.

[Hughes and Foley, 2014] Hughes, J. F. and Foley, J. D. (2014). *Computer graphics: principles and practice*. Pearson Education.

[Kaiming, 2015] Kaiming, H. (2015). Convolutional feature maps. elements of efficient (and accurate) cnn-based object detection.

[Kaufman, 1993] Kaufman, A. (1993). *Rendering, visualization and rasterization hardware*. Springer Science & Business Media.

[Lauer *et al.*, 2007] Lauer, F., Suen, C. Y., and Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6), 1816–1824.

[LeCun, 1998] LeCun, Y. (1998). The mnist database of handwritten digits. [Online; accessed 05 April 2018]http://yann. lecun. com/exdb/mnist/.

[LeCun *et al.*, 1998a] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[LeCun *et al.*, 1998b] LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998b). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–50). Springer.

[Maas *et al.*, 2013] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30 (pp.$\tilde{3}$).

[Marr and Hildreth, 1980] Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proc. R. Soc. Lond. B*, 207(1167), 187–217.

[Nielsen, 2015] Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press.

[Opitz and Maclin, 1999] Opitz, D. W. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *J. Artif. Intell. Res.(JAIR)*, 11, 169–198.

[Ramachandran *et al.*, 2018] Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions.

[Rogers and Girolami, 2016] Rogers, S. and Girolami, M. (2016). *A first course in machine learning*. CRC Press.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3), 210–229.

[Schantz, 1982] Schantz, H. F. (1982). *History of OCR, optical character recognition*. Recognition Technologies Users Association.

[Scherer *et al.*, 2010] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks* (pp. 92–101).: Springer.

[Shiffman, 2012] Shiffman, D. (2012). *The Nature of Code: Simulating Natural Systems with Processing*. Daniel Shiffman.

[Simard *et al.*, 2003] Simard, P. Y., Steinkraus, D., Platt, J. C., *et al.* (2003). Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3 (pp. 958–962).

[Singh *et al.*, 2017] Singh, S., Paul, A., and Arun, M. (2017). Parallelization of digit recognition system using deep convolutional neural network on cuda. In *Sensing, Signal Processing and Security (ICSSS), 2017 Third International Conference on* (pp. 379–383).: IEEE.

[Smith, 2013] Smith, S. (2013). *Digital signal processing: a practical guide for engineers and scientists*. Elsevier.

[Steinkraus *et al.*, 2005] Steinkraus, D., Buck, I., and Simard, P. (2005). Using gpus for machine learning algorithms. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on* (pp. 1115–1120).: IEEE.

[Suzuki *et al.*, 1985] Suzuki, S. *et al.* (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1), 32–46.

[Szeliski, 2010] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

[van Rossum *et al.*, 2001] van Rossum, G., Warsaw, B., and Coghlan, N. (2001). Pep 8: style guide for python code. *Python. org*.

[Wong *et al.*, 2016] Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on* (pp. 1–6).: IEEE.

[Zeiler, 2012] Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Appendix A

Projnet Architecture in Python

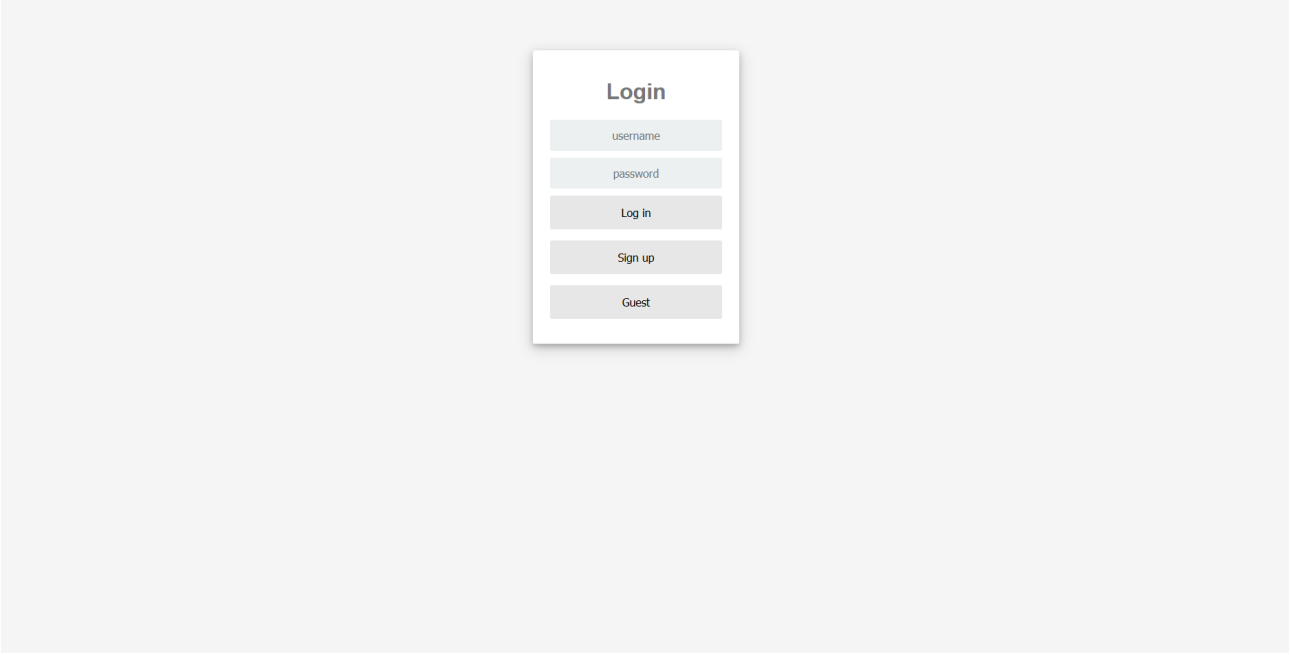
```
def create_lenet5(training_data, width=28,height=28):
    '''Create a lenet-5 inspired network
    Creates a neural network which uses two convolutional layers and two
    pooling layers. This is the architecture for the netowork used in
    experiments carried out in the project.
    Args:
        training_data: the training data, used to tailor the network to
        the data.
        width: 28 pixels
        height: 28 pixels

    Returns:
        model: a sequential model of a neural network
    '''
    (test_img, test_label), (train_img, train_label), mapping, number_classes = training_data
    input_shape = (height, width, 1)
    number_filters = 32   # number of convolutional filters to use
    pool_size = (2, 2)    # size of pooling area for max pooling
    #kernel_size = (3, 3) # convolution kernel size
    model = Sequential()
    #Convolutional input Layer.
    #Takes an input vector of size (28,28,1) so 784 inputs
    model.add(Convolution2D(12, (5, 5), activation = 'relu',
                                        input_shape=input_shape,
                                        init='he_normal'))
    #Max pooling layer using pool_size (2,2)
    model.add(MaxPooling2D(pool_size=pool_size))
    #Convolutional layer
    #25 filters with kernel size of (5,5)
    model.add(Convolution2D(25, (5, 5), activation = 'relu',
                                        init='he_normal'))
    #Max pooling layer using pool_size (2,2)
    model.add(MaxPooling2D(pool_size=(2, 2)))
    #Flatten to 1d tensor
    model.add(Flatten())
    model.add(Dense(180, activation = 'relu', init='he_normal'))
    model.add(Dropout(0.5))
    model.add(Dense(100, activation = 'relu', init='he_normal'))
    model.add(Dropout(0.5))
    #Output layer
    #outputs are the number of classes in the data set
    #Softmax to squash all values between 0 and 1
    #initialize weights using he_normal
    model.add(Dense(number_classes, activation = 'softmax', init='he_normal'))
        model.compile(loss='categorical_crossentropy',  optimizer='adamax',
                                                        metrics=["accuracy"])
    return model
```

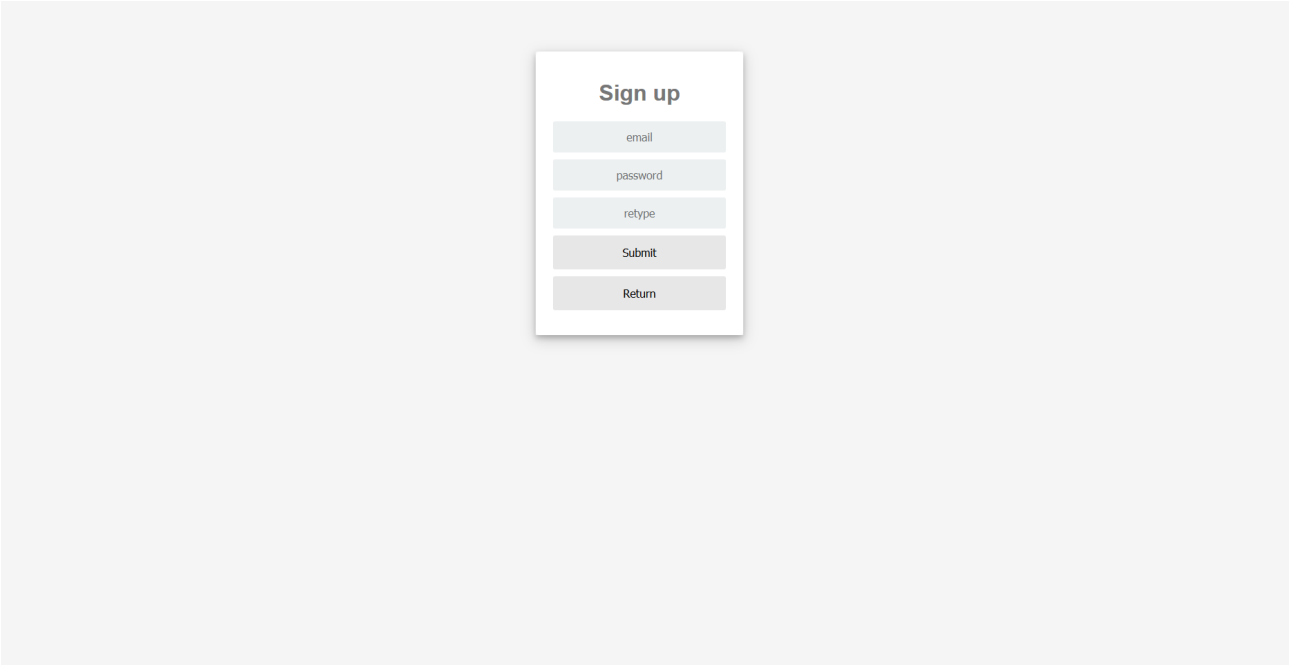
Appendix B

Screen Shots of Implementation

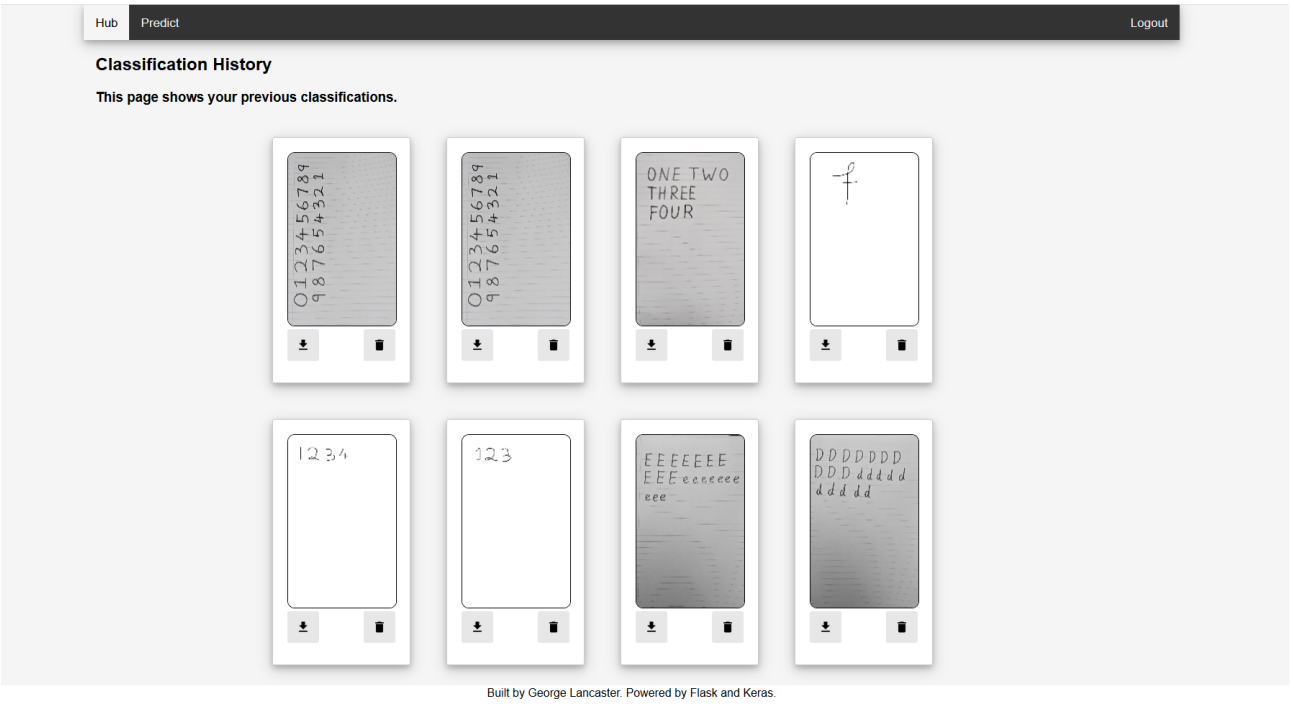
B.1 Login Page



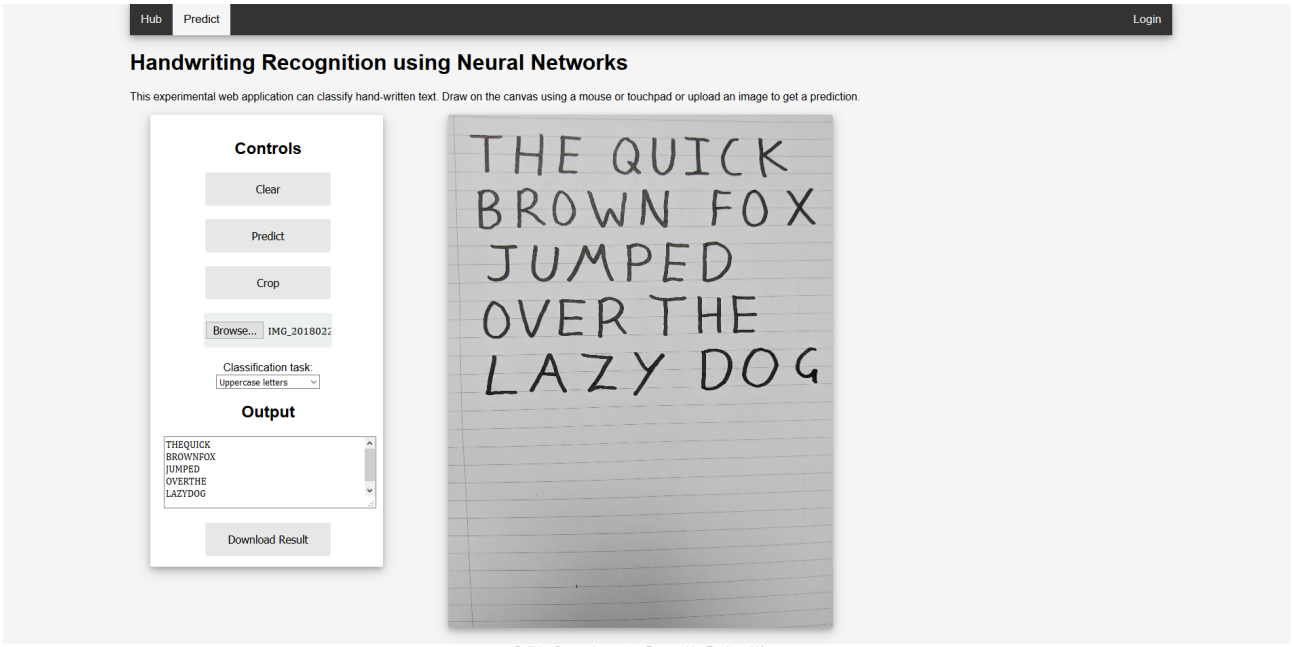
B.2 Create User Page

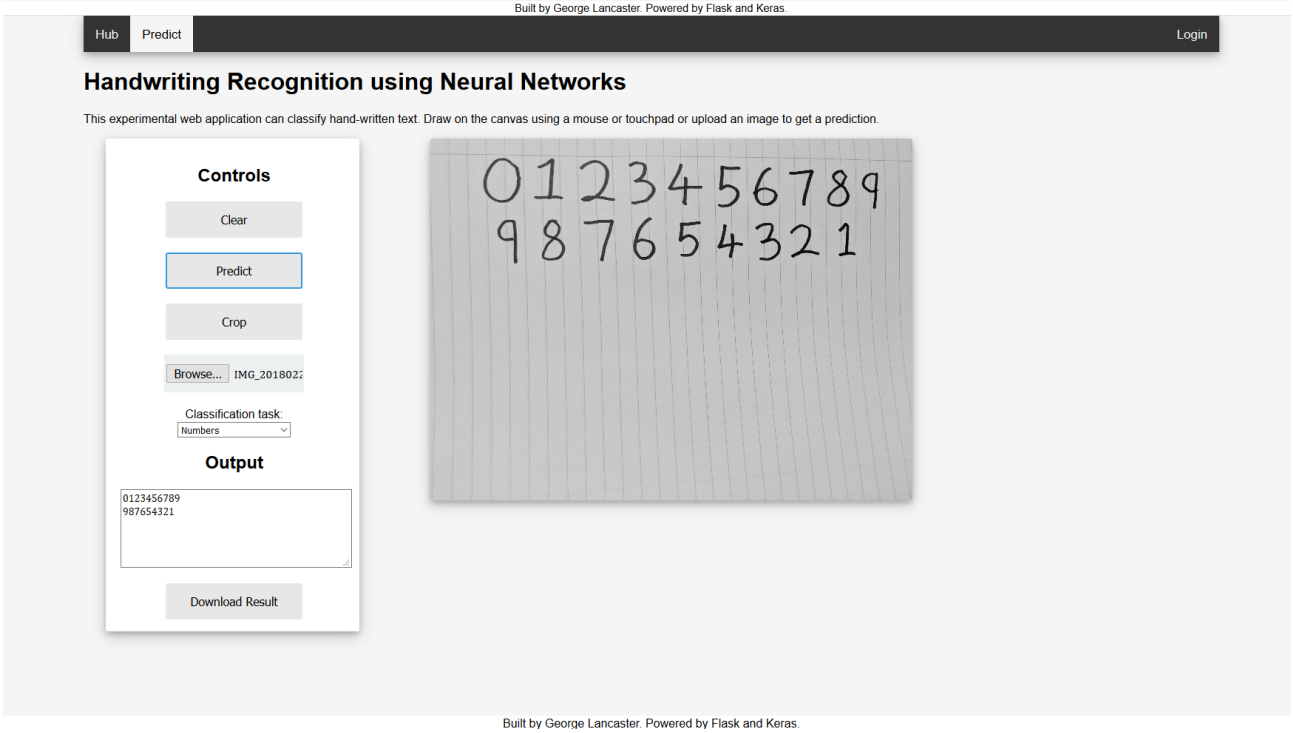


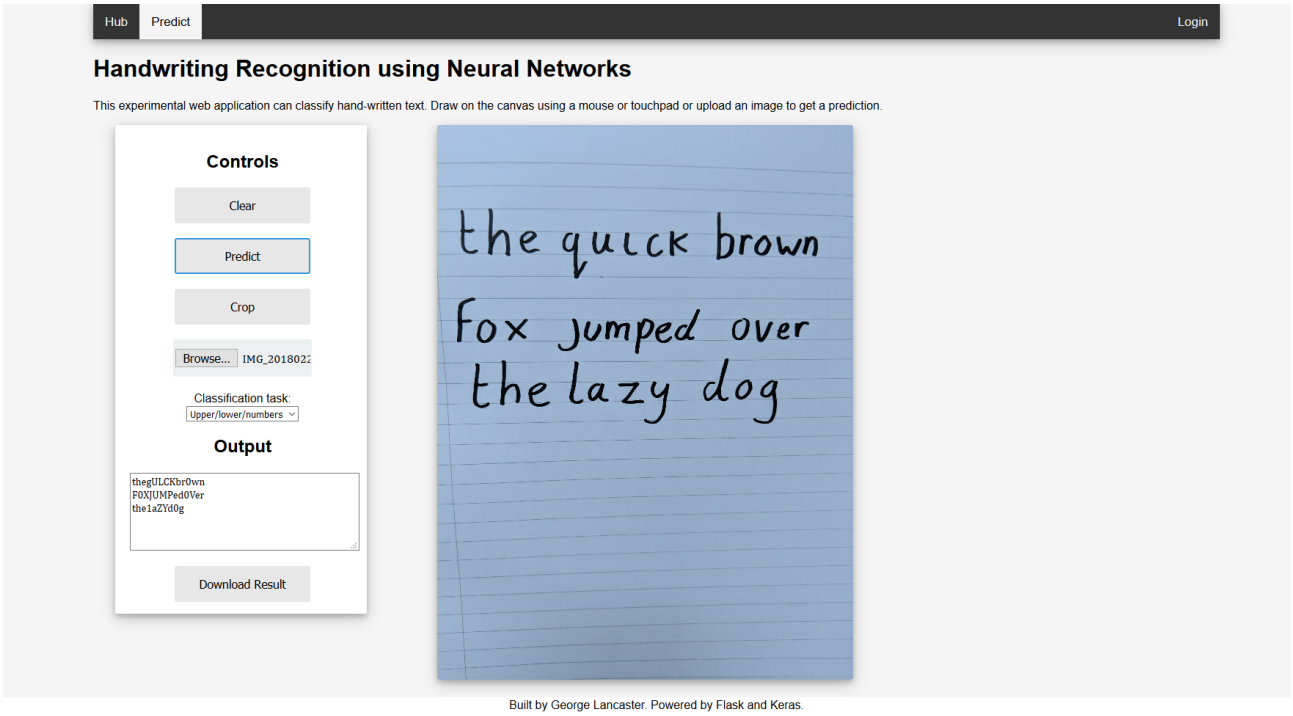
B.3 Hub Page



B.4 Classify Page






Hub
Predict
Login
Handwriting Recognition using Neural Networks
This experimental web application can classify hand-written text. Draw on the canvas using a mouse or touchpad or upload an image to get a prediction.
Controls
Clear
Predict
Crop
Browse...
IMG_2018022
Classification task:
Upper/lower/numbers
Output
thegULCKbr0wn
F0XJUMPed0Ver
the1aZYd0g
Download Result
the quick brown
fox jumped over
the lazy dog
Built by George Lancaster. Powered by Flask and Keras.

Appendix C

Work Diary

```
10/10/17 ------------------------------------------------------
Looked at deeplearning4J, using the MNIST dataset for image classification.
One of the biggest problems with teaching a neural network is the amount of data
 they need to learn. I may have solved this issue for my project by using a
 dataset already well-known in the deep-learning community by using a dataset
 called MNIST.
 I have used the example from the deeplearning4J as a feed-forward network. In
 this example, there is a single layered network which classifies images from
 the MNIST dataset as numbers (0-9).

12/10/17 ------------------------------------------------------
Played around with Neuroph studio. Didn't make any significant progress.

13/10/17 ------------------------------------------------------
Have got DL4J working in my own java project in eclipse. Have used the code
which generates the output above in my own project. I can now use this to start
creating a prototype.Need to look into pretrained and persistent networks to
stop from training the network every time I debug. I will begin to create a
GUI which takes an input and tests it against the trained network, guessing
the number.

31/10/17 ------------------------------------------------------
Using a CNN in java to classify my own set of images. Sets of numbers from 0-3.
After speaking with Stelios, I have decided that python will be more suitable
for this project.

01/11/17 ------------------------------------------------------
Set up python on my laptop. I have also set up keras and tensorflow.

10/11/17 ------------------------------------------------------
# FROM 01/11/17 to 10/11/17
Re-wrote my Java convolutional network using Keras. I then trained the net on
the MNIST dataset. Followed tutorials online on how to do this, there are a lot
of resources online for this.

11/11/17 ------------------------------------------------------
Created a simple GUI to test the input of my own data. This is only single
digits.
```

```
14/11/17 ----------------------------------------------------
Started experimenting with different configurations of networks.

25/11/17 ----------------------------------------------------
I have found a new data-set called EMNIST (the extended MNIST). It contains
letters as well as numbers. I trained my convolutional network with upper-case
 letters.

03/12/17 ----------------------------------------------------
I have been able to extract numbers from images and classify them in order.
This is still very buggy; however, I have a 100% match with some of my test cases.
In order for this to work, numbers must be double-spaced so as not to confuse
the extraction process. I have used a mixture of blob FE and others.

05/12/17 ----------------------------------------------------
I have applied the same logic to letters, however the classification accuracy
is low.

08/11/17 ----------------------------------------------------
Looking to host it as a web-app to make it more accessible. Choices include

  1. Django
  2. Flask

Will play around with these to see what fits better, At the moment, Django
looks good.
12/12/17 ---------------------------------------------------- 7 hours
1. Get the image cropping to work properly.
  1.1 Crop a single digit from an image     #Done
  1.2 Crop a string of digits from a word   #Done
  1.3 Crop a sentence from a paragraph      #Done
  1.4 Crop a paragraph from a page          #Not sure if needed
  1.5 Classify a single digit               #Done

2. Decide what level of cropping is required for an image
  2.1 Single digits
  2.2 Sentences
  2.3 Paragraphs

I have three options for this:
  1. I can train a classifier to decide if an image fits these parameters
or
  2. I can have a set of criteria that an image needs to fit to be classified
    as either a digit, sentence or paragraph
  3. Perform all operations on all images

End of day:

I can crop sentences, paragraph and single digits for classification.
It is still buggy, however when classifying fonts, accuracy is 99%.
Improved the paint GUI to crop input images.

16/12/17 ----------------------------------------------------6 hours
TODO
```

1. Create a better looking GUI to include all of the functionality achieved
in the last session. #Not done
2. Investigate hosting online further. #Done
3. Iron out any bugs found so far. #Done

End of day:
I have created a simple web application which takes the networks I
have created and classifies strings and single digits using them.
There is a bug preventing the results from being displayed on the app,
however the classification accuracy is good.

Next time I want to experiment with classifying each digit without breaking
the image into lines beforehand. To do this I will need to order each of the
detected digits from their position from the top left hand corner of the screen.
I may have to devise an algorithm to decide the order of classification.

TODO:
  1. Upload images / pdf files for conversion
  2. experiment with other methods of digit detection

17/12/17 ---------- 2 hours

  1. Experiment with other classification methods #Done, got another method
     nearly working but cannot order the contours correctly due to slight
     differences in the y value.
  2. Upload images

22/12/17 ---------- 4 hours

Made some aesthetic changes to the web application. Added the ability to
upload images.

24/12/17 ---------- 4 hours

Attempted to train a network using 697932 data items (upper-case, lower-case
and numbers 0-9). A problem is that my laptop does not have a powerful enough
GPU to train quickly. Currently, it would take over 200 hours to train the
network with 200 epochs.

I have enquired with the university about getting access to a render farm,
however I have not yet heard back.

27/12/17 ---------- 4 hours
Made some changes to the web application. Added controls for adding own
images to the canvas.

29/12/17 ---------- 6 hours

I have purchased a computer with a dedicated graphics card to solve the problem
I had on 24/12/17. Training time is now down to ~2 minutes/epoch, so it is
reduced by 90%. This gives me much greater flexibility as I can train many
networks using different characteristics to compare them.

Done today:
  1. Trained a network for 62 classes (upper, lower, numbers)

  TODO:
  1. Test accuracy

31/12/17 ---------- 2 hours

Network trained on 29/12/17 classifies all inputs as 'M'. Need to change the number of epochs to a lower number as I think the network may be over-fitted.

06/01/18 ----------

Cropping algorithm is nearly complete, works pretty well on most test cases.

Two errors which need fixing :
  1. Sometimes sorts digits from string in the wrong order, this is due to the height at which they appear on screen where higher = sorted first.

  2. Inputs "1", "l" and "i" are sometimes cropped wrong when written in a certain style as they are too thin. One way to fix this could be to check the thickness of a contour and extend the cropping area, thus preserving the shape of the character.

TODO:
  1. Train some networks, try to improve classification.

End of day:
  1. Have trained a recurrent neural network with very good accuracy (~70%) for all cases (letters, numbers).
  2. I have started to use real hand-written images, rather than ones drawn in paint for classification.

08/01/18 ----------

Just played around with training new networks:
  1. Number recognition, 20 epochs, with a recurrent network.

10/01/18 ---------- 3 hours

Added the ability to select which type of network is used in the web application through a drop-down menu.

This was fairly simple to implement, I had to create a new method to change the network loaded in main, then link this up to the JavaScript to send and retrieve the type of classification task.

11/01/18 ---------- 5 hours

Trying to fix problem by where the order of the letters are read top to bottom
rather than left to right. The idea is to rotate the input image by ~5 degrees
then rotate the individual letters back 5 degrees.

This could break the line classification when all characters are the same
height.

13/01/18 ----------
Things to do:

Must:
1. Improve classification accuracy.
2. Format textual output.
3. Resizable canvas.

Should:
1. Create user logins to save predictions.

14/01/18 ---------- 4 hours
Need to allocate positioning of characters on screen to try to preserve
formatting on output.

Also, need to add an eraser tool, will be the same as drawing tool but set the
colour to white.

To improve the classification accuracy of some words, I may implement a spell
checker to recommend replacement words.

16/01/18 ---------- 5 hours
Have continued development on the web-application.

I will be creating user accounts etc to store images and previous classifications.

17/01/18 ---------- 7 hours
Created a database and user access. Users' can login. The rest of the app
is not currently personalised, however I will be making these changes shortly.

18/01/18 ---------- 8 hours
Made changes to the sign-up and database, as well as trying to write a new
page segmentation algorithm.

20/01/18 ---------- 5 hours
Mostly aesthetic changes. Updated the sign-up, classify and hub pages.
Added a back button to the sign-up page. Working on adding the ability to save
and download classifications. I have added all the buttons, just need to
include functionality.

The hub page now displays all of a users' previous classifications.

22/01/18 ---------- 2 hours
Started working on creating a universal training module.

1. create a model
2. train the model
3. use tensorboard callbacks

23/01/18 ---------- 7 hours

Almost finished universal training module.

Just need to add in more networks.

TODO: Try flipping all bits in classification image, see if that improves
accuracy.

24/01/18 ---------- 4 hours
Finished a new page segmentation algorithm for getting text from images.
This now preserves line formatting on output.

The algorithm works by sorting the contours into lines first, and their order
within the line second. There is still no method for creating

25/01/18 ---------- 1 hour

30/01/18 ---------- 2 hours
Made changes to train_network

02/02/18 ----------

14/02/18 ---------- 5 hours
Have been very busy over the past two weeks, haven't had much time to work on
the project. I am now working on building a module to test networks, and draw
graphs of their effectiveness. Confusion matrices, etc.

I have completed this today, output graphs are looking good.

15/02/18 ---------- 6 hours
Started working on creating a test for the character extraction. So far, only
label contours from left-to-right-top-to-bottom. Need to devise a test to show
that MOST inputs get labelled correctly.

From here on out I will mostly be devising tests to show the effectiveness of
the networks and algorithms I have created thus far.

16/02/18 ---------- 9 hours
Played around with training some networks. I have also spent some time
generating graphs to be used in the report. I have laid out a rough template
for the report and I have begun writing a bit about image transformations.

21/02/18 ----------
Continue work on the report made changes to the confusion matrices for testing
the networks.

22/02/18 ----------
Begun writing more about the feature extraction process. Contour detection etc.
Want to get a solid draft for this section completed before writing anything

else. Am going away for the weekend so will not be able to work on this until
at least the 26th.

23/02/18 ----------
Not leaving until later so have a few hours to work.

27/02/19 ---------- 2 hours
Have implemented a version of the lenet-5 network. This has given a massive
increase in accuracy, has almost made the old network redundant.

28/02/18 ---------- 3 hours
Continued research into why my model is over-fitting, but it is not showing
up on the graph. Maybe I am testing something wrong.

Spoke to Stelios today, showed him the deliverable. He thinks that it is done,
and that I should have a draft report complete in 2 weeks. Will try and get it
to a fairly good standard by then, but still have a lot of analysis to do!

01/02/18 ---------- 5 hours
I am working on creating my own dataset to test the network on. This will be
more accurate than testing on part of emnist because my dataset consists of images
which have been extracted using my algorithm, and therefore give a better
representation of the accuracy of the task.

To continue where I left off:
1. generating test data with create_test_data.py
2. evaluating the model by messing around with code in area.py

02/02/18 ---------- 4 hours
I have generated a dataset of 757 hand written images, I can evaluate the Model
and get >90% success rate, which is good! Just need to figure out how to plot
the new data in a confusion matrix like I have done with the old data. I am sure
this it is possible, however keras's ImageDataGenerator class does not make it
easy, as it processes the images in batches. It may be easier to generate my
own method for using the data.

The problem here is that the classes are given in a list, rather than an np array
of one-hot-encoded labels.

04/03/18 ---------- 7 hours
Massive overhaul to the database for collecting and retrieving user predictions.
Now, each image has its own location stored on the database and each prediction.
Can *almost* download a prediction on a button click. Would like to be able
to do this as a text file.

Nearly there with the application, feel like it is almost finished, just needs
cosmetic changes.

07/03/18 ----------
Can now download the predictions from the hub. Continued the report and started
comparing the results from my experimentation.

Think I will need some help with analysing the data as I am not sure if the
validation accuracy or validation loss is more important.

12/03/18 ----------
Resumed work on the report. Struggling to find a good structure to show the
various experiments I have done.

15/03/18 ----------
Continued work on report.

16/03/18 ----------
Continued work on report. Have made good progress on the Background section.

19/03/18 ----------
Have almost finished the background section of the report and I have made a
start of all other sections. The report is currently 29 pages long. 4500 words.

21/03/18 ----------
Created a graphic to show a selection of incorrect classifications. This shows
that the network is well-trained as many of the wrongly classified characters
are genuinely ambiguous.

22/03/18 ----------
Continued working on the report. I have had a minor breakthrough with training
with image transformations as it has set a new current record for best
classifier.

23/03/18 ----------
Continued work on report, created networks 2.1 - 2.7. and tested them using
emnist-letters. Error % is very low.

27/03/18 ----------
Wrote up the experiments, created graphs and filled out the table.
Also proof read parts of the report.

28/03/18 ----------
Tested the networks against my PDS data set. Results are pretty bad apart
from one network.

29/03/19 ----------
Tried to make changes to the char-extractor. Didn't make much progress.

31/03/19 ----------
Changed the char extractor to keep the proportions of the extracted chars,
rather than just resizing to 28x28. This *should* improve accuracy of the
web application.

02/03/19 ----------
Continued work on report, which is currently on 7500 words.

04/03/19 ----------
Continued work on report. Proof read it up to chapter 5. Added in bits
which were not complete. Also re-wrote poorly written parts.
Word count is between 8000-8700 according to montery languages word counter.
Change et al to italics.

05/03/19 ----------
Added more to the reflection. Need to include my gantt chart

08/04/18 ----------
Made aesthetic changes to the app. Changed colour scheme and layout.

Also added guest users. Still need to sort out creating a new user as
adding guest users has broken this.

10/04/18 ----------
More work on the report, added more detail to background.

Need to add detail to backpropagation algorithm. Currently is a bit basic.

11/04/18 ----------
A few changes to the app, login and database.

12/04/18 ----------
Added to background for the backprop algorithm and more detail on
conv nets.

Need to fully document code as it is currently a bit messy.

13/04/18 ----------
Documenting code.
  Adding comments to all methods to make it readable;
  Changing some variable names to be clearer;
  Removing un-needed object declarations.
Need to finish verification on login.

14/04/18 ----------
Played with ensemble classifiers but didn't get significant enough results
to include in report.

16/04/18 ----------
Realised today that my results are wrong for the last experiment. The
accuracy is much higher than I thought. It is actually >95% accurate. Very
happy as this proves that the networks can be generalised to real-world
problems. The problem was that the custom testing data had not been
normalised.

17/04/18 ----------
Re-tested all the classifiers using the new normalised data. Now have new
Values for the experiments. Have added these in to the report, but need
to make changes throughout the entire report.

18/04/18 ----------
Reading through from start to finish, currently on page 10.

19/04/18 ----------
Continued to read through, on page 15.

21/04/18 ----------
Uploaded the application to the internet on
georgelancaster.pythonanywhere.com

Predictions do not work, could be something to do with multithreading.

Can access the site but I am getting a 499 error.

22/04/18 ----------
Tried to fix the web hosting, spent all day trying to fix it.

Continued to read through report, currently on page 20.
Change diagrams from squares to circles.

23/04/18 ----------
Continued to read through report and tidied up some code in train_network.py.
Have done so, so that it can be run just using train_network.py without
providing any parameters.
On page 24

24/04/18 ----------
Finally got the application to work on georgelancaster.pythonanywhere.com

The problem was that errors occur when tensorflow models are accessed in
a different thread than the one they are created in. I solved this issue
by changing the Keras back-end to Theano on the web-site.

I have kept the back-end as TensorFlow for the model I am submitting.

on page 26.

25/04/18 ----------
Read through some more of the report, now on page 32.

26/04/18 ----------
After showing the web application to some people, found that image
cropping would be a good idea to make it more usable.

Have implemented a cropping tool on the classification page.

28/04/18 ----------
Changes some of the figures to tables changed some wording. and changed
some of the formatting.

Need to continue from page 32.

29/04/18 ----------
Trained the final versions of the classifiers in the web-application.

Also continued reading to page 36.

01/05/18 ----------
Cleaned up some of the code and finished proof-reading

04/05/18 ----------
Cleaned up all of the code, formatted all modules and quickly proofread
report.

05/05/18 --
Proof-read report, cleaned up the contents, list of figure, tables.
Also added in two more figures, bar chart for classifier comparison and
misclassifications for EMNIST-digits. Also made a new cover page.

Project is pretty much ready for submission, just need to proof read once
more.

06/05/18 --
Proof read some more.
Need to update screenshots. Will try and print off tomorrow and submit.

EOF

Appendix D

Python Requirements

```
bleach==1.5.0
click==6.7
cycler==0.10.0
enum34==1.1.6
Flask==0.12.2
Flask-Login==0.4.1
Flask-SQLAlchemy==2.3.2
Flask-WTF==0.14.2
h5py==2.7.1
html5lib==0.9999999
itsdangerous==0.24
Jinja2==2.10
-e git://github.com/fchollet/keras.git@eac78b859beb31cafa65a3edb4eaa888d3b6c2e6#egg=Keras
Markdown==2.6.11
MarkupSafe==1.0
matplotlib==2.1.1
numpy==1.13.3
opencv-python==3.4.0.12
Pillow==5.0.0
protobuf==3.5.1
pyparsing==2.2.0
python-dateutil==2.6.1
pytz==2017.3
PyYAML==3.12
scikit-learn==0.19.1
scipy==1.0.0
six==1.11.0
SQLAlchemy==1.2.1
tensorflow==1.4.0
tensorflow-tensorboard==0.4.0rc3
Werkzeug==0.14.1
wincertstore==0.2
WTForms==2.1
```

Appendix E

Training Speeds

E.1 Training using CPU

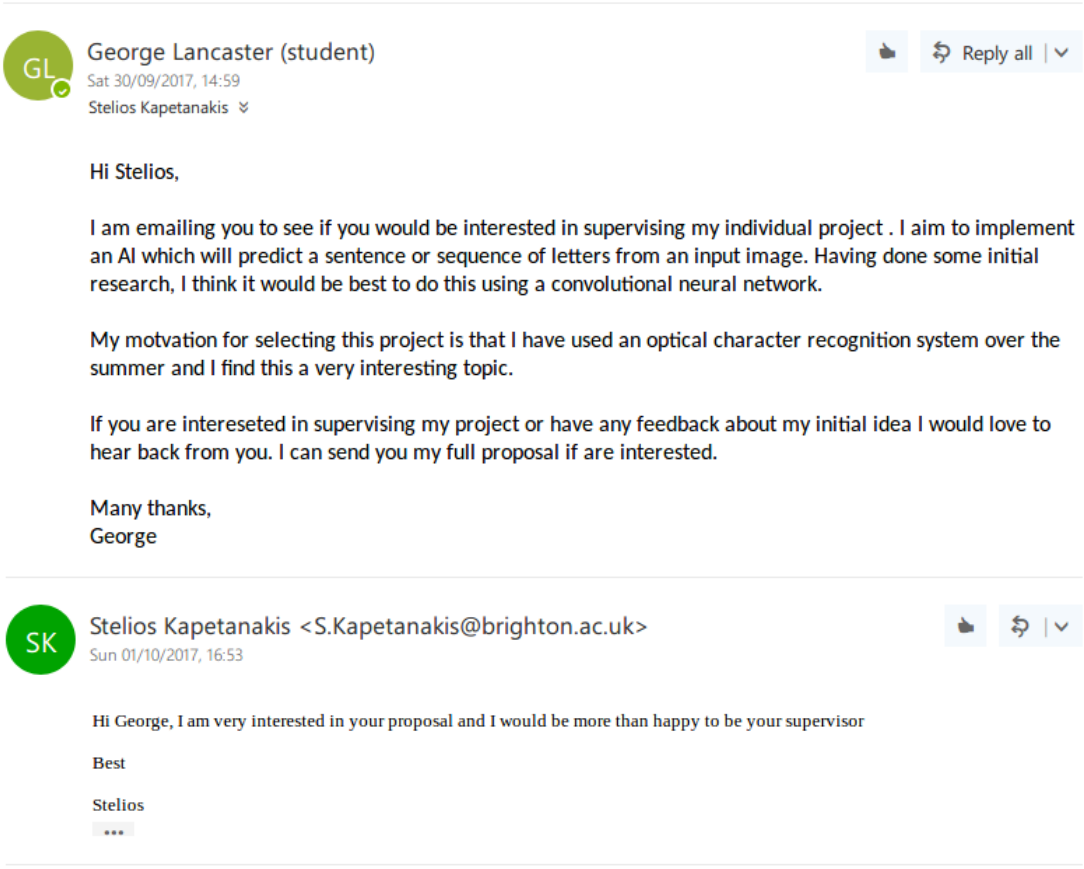
```
Train on 124800 samples, validate on 20800 samples
Epoch 1/1
 62208/124800 [=============>.................] - ETA: 24s - loss: 2.1358 - acc: 0.3702
```

E.2 Training using GPU

```
Train on 124800 samples, validate on 20800 samples
Epoch 1/1
 64000/124800 [==============>...............] - ETA: 7s - loss: 2.1879 - acc: 0.3616
```

Appendix F

Supervisor Correspondence



GL George Lancaster (student)
Sat 30/09/2017, 14:59
Stelios Kapetanakis

Reply all

Hi Stelios,

I am emailing you to see if you would be interested in supervising my individual project . I aim to implement an AI which will predict a sentence or sequence of letters from an input image. Having done some initial research, I think it would be best to do this using a convolutional neural network.

My motvation for selecting this project is that I have used an optical character recognition system over the summer and I find this a very interesting topic.

If you are intereseted in supervising my project or have any feedback about my initial idea I would love to hear back from you. I can send you my full proposal if are interested.

Many thanks,
George

SK Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>
Sun 01/10/2017, 16:53

Hi George, I am very interested in your proposal and I would be more than happy to be your supervisor

Best

Stelios

GL George Lancaster (student)
Wed 18/10/2017, 13:11
Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Reply all

Hi again Stelios,

Do I need to include an ethics form in my proposal? If so, can I meet with you for you to sign mine?

Many thanks,
George

SK Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>
Wed 18/10/2017, 18:18

Hi George,

Let's have blank the ethics fields. Just have my name in capital in the supervisor's name since I am more than happy to supervise this project

Best regards

Stelios

GL George Lancaster (student)
Sat 28/10/2017, 11:17
Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Reply all

Hi Stelios,
Are you available to meet any time next week to discuss my project further?
I can be free any day of the week.
Many thanks,
George

SK Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>
Tue 31/10/2017, 15:10

Let's have a chat around 12.15. Let's meet in Cockcroft mezzanine cafe.

Best

Stelios

GL George Lancaster (student)
Sun 12/11/2017, 16:31
Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Reply all

Hi Stelios,

I can meet with you at 11am on tuesday if that is okay with you?

Thanks,

George

SK

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Sun 12/11/2017, 19:53

George Lancaster (student)

Reply all

Hi George,

Yes, that is absolutely fine with me

Best

Stelios

Dr. Stelios Kapetanakis
Principal Lecturer in Business Intelligence and Enterprise

...

SK

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Mon 20/11/2017, 23:34

Constantinos Ioannou (student); George Lancaster (student); Consuelo Levinta (student); +1 more

Reply all

Hi guys,

Haris (project second reader) can make it next Tuesday between 13.00-15.00. Would that be a good time for you?

Please feel free to "reply all" to this email. Once I have your answers I will send the final slot arrangement(s) (it will be ~15-20 minutes each).

Best regards

Stelios

GL

George Lancaster (student)

Tue 21/11/2017, 00:11

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>; +4 more

Reply all

Hi Stelios,

I can do any time between 2pm and 3pm that day.

Thanks,

George

SK

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Thu 23/11/2017, 00:38

Dear all,

Thank you so much for arranging this so smoothly.
Our final timeslots for Tuesday, 28th of November are the following:

13.40 - 14.00	Constantinos
14.00 - 14.20	Consuelo
14.20 - 14.40	George
14.40 - 15.00	Harry

Best regards

Stelios

SK

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Tue 28/11/2017, 21:03

George Lancaster (student); Haris Mouratidis

Reply all

Hi George,

Please find your viva feedback below.

Best

Stelios

GL

George Lancaster (student)

Mon 15/01, 15:13

Stelios Kapetanakis

Reply all

Action Items

Hi Stelios,

Sorry, I have been very busy the past month. I hope you had a good new year.

Would it be possible to arrange a meeting with you about my project and the future!

Please let me know a time/date you would be free.

I have also submitted my application to UoB today, and would be very grateful if you could send off my reference.

Many thanks,

George

SK

Stelios Kapetanakis <S.Kapetanakis@brighton.ac.uk>

Mon 22/01, 12:54

Harrys Pitsillides (student); George Lancaster (student)

Reply all

Hi Harry, George,

I would like to invite you to Gluru AI for this Thursday at 1.30-2.30pm.
During that time we will have the opportunity to show you around and talk about your projects.

The address is We Work Aldwych House, 71-91, Aldwych, London WC2B 4HN

Dr. Stelios Kapetanakis
Principal Lecturer in Business Intelligence and Enterprise

SK

Stelios Kapetanakis

Wed 24/01, 14:09

harrys pitsillides <harryspitsillides@hotmail.com>; George Lancaster (student); +1 more

Reply all

Harry, George,

Would it be ok to shift our meeting tomorrow an hour earlier? 12.30pm

I am afraid I will need to be in a teleconference call at 2pm.

Please confirm whether this would be ok with you

Best

Stelios

GL

George Lancaster (student)

Wed 24/01, 14:41

Stelios Kapetanakis

Reply all

Hi Stelios,

Yes that time is fine for me.

Many thanks,

George

Sent from Mail for Windows 10

GL

George Lancaster (student)

Wed 14/02, 16:55

Stelios Kapetanakis

Reply all

Hi Stelios,

I have pretty much finished the implementation of my project and I am now writing the documentation, I just have some questions regarding comparing the success of networks.

I have written a few basic functions to draw confusion matrices of the results of large tests, however one of the networks has 62 possible outputs (upper-case letters, lower-case letters and digits 0-9). Are there any other graphical representations you could recommend I investigate as the graphs I have generated seem difficult to read.

Many thanks,

George

SK

Stelios Kapetanakis

Thu 15/02, 15:16

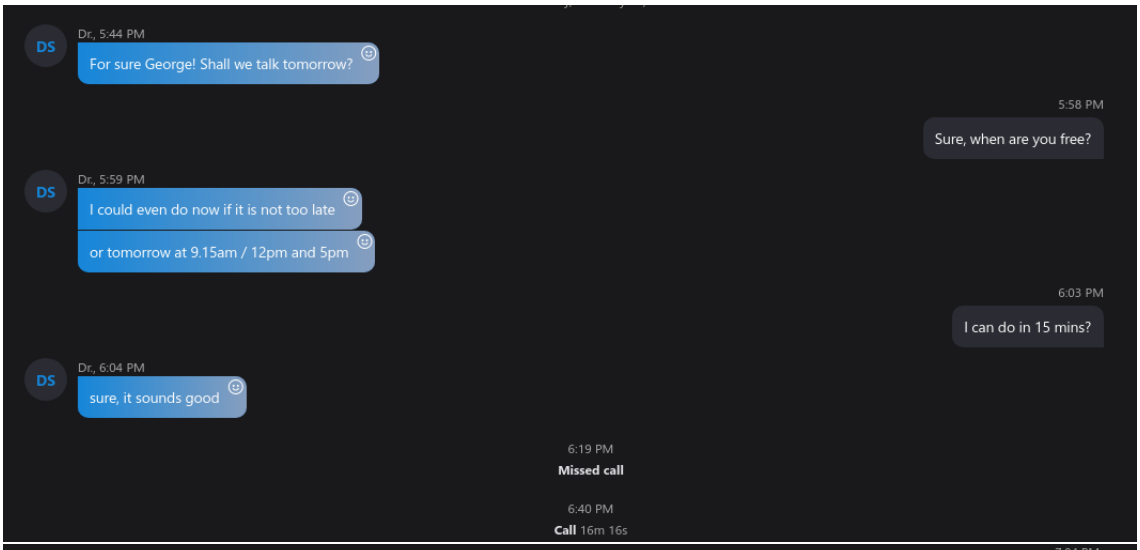
George Lancaster (student)

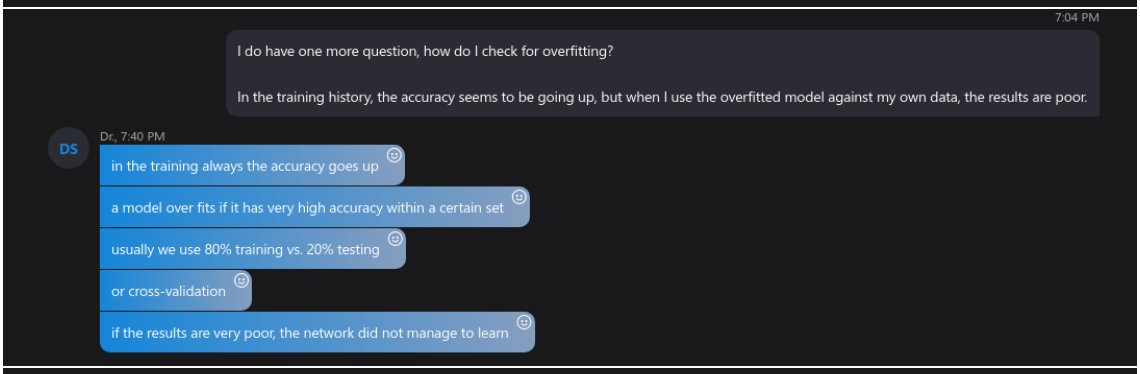
Reply all

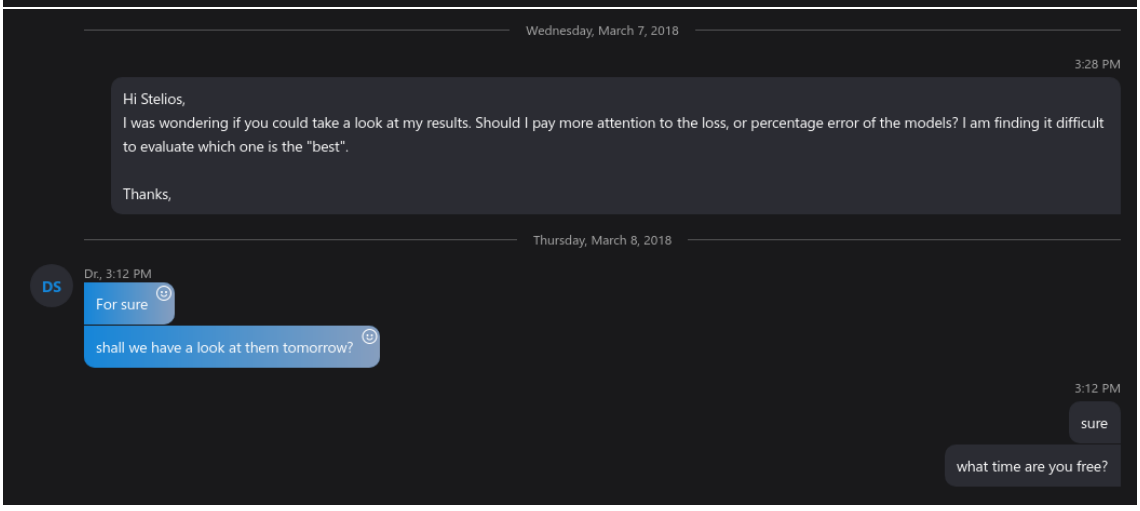
Small graphs may be better since they focus on a few outputs at a time
or if you have too many components you can use PCA: Principal Component Analysis

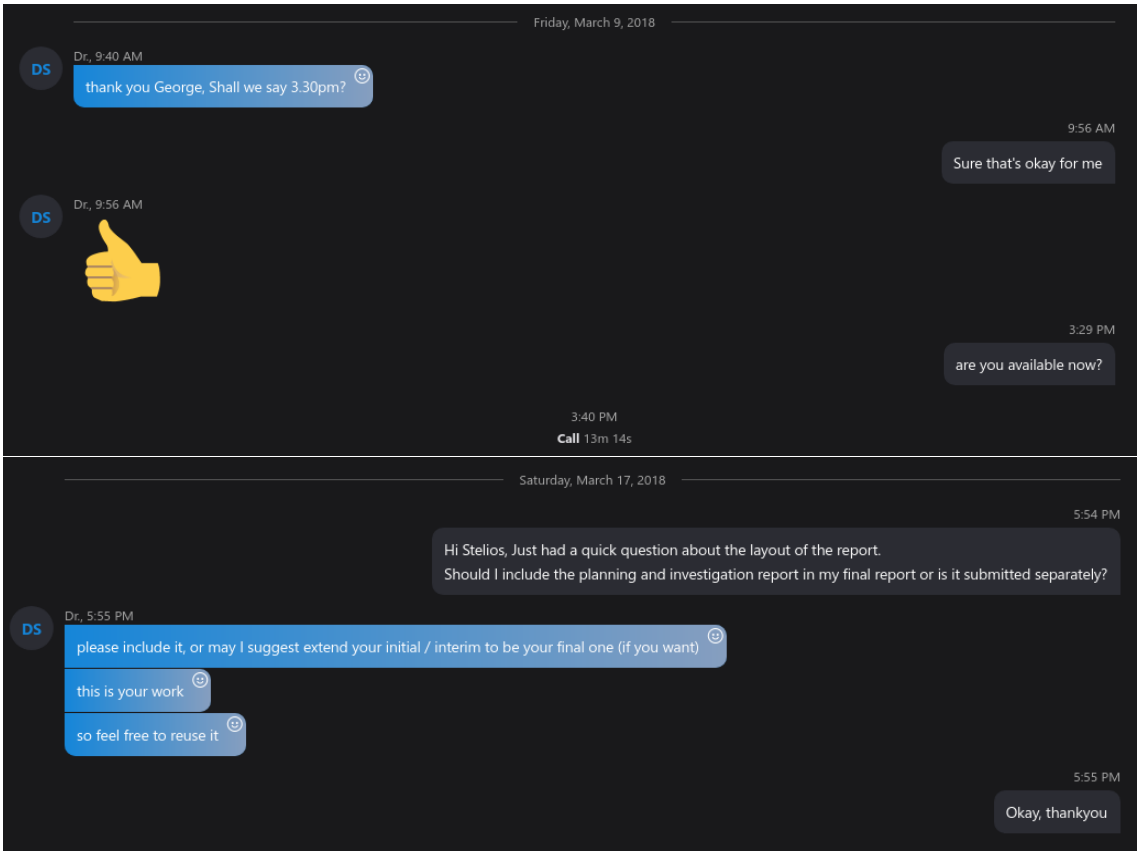
Best regards

Stelios








Friday, March 9, 2018
DS
Dr., 9:40 AM
thank you George, Shall we say 3.30pm?
9:56 AM
Sure that's okay for me
DS
Dr., 9:56 AM
3:29 PM
are you available now?
3:40 PM
Call 13m 14s
Saturday, March 17, 2018
5:54 PM
Hi Stelios, Just had a quick question about the layout of the report.
Should I include the planning and investigation report in my final report or is it submitted separately?
DS
Dr., 5:55 PM
please include it, or may I suggest extend your initial / interim to be your final one (if you want)
this is your work
so feel free to reuse it
5:55 PM
Okay, thankyou

Appendix G

Ethics Form

SCHOOL OF COMPUTING, ENGINEERING & MATHEMATICS ETHICS FORM

This ethics form is designed to help you quickly and easily identify how you should approach any ethical issues raised by your project or dissertation. It should be completed for ALL research projects and dissertations prior to the commencement of the project. Please do not approach any participants involved in the research until this have been completed and discussed with your supervisor or member of the CEM ethics committee (if appropriate).

This form must be completed by the project student or researcher responsible for the project. Once completed, you should discuss it with your supervisor to ensure that you take the right follow-up actions.

If you answer 'No' to all questions in this form and this is confirmed with your supervisor (if appropriate) then no further action is required. Please note that in signing this form you accept that it is still your responsibility for your project or dissertation module to follow the **University's Guidance on Good Practice in Research Ethics and Governance**, available on StudentCentral. Any significant change in the question, design or conduct of your project or dissertation that would alter your answers on this form must be notified to your supervisor who will advise you on whether you need further action.

If you have answered 'yes' to *any* of the questions in Section B of the Student Checklist your supervisor will need to make a judgment as to whether or not the research includes more than a minimum level of risk. If this is the case then your supervisor will need to email this form to the CEM ethics committee (CEMethics@brighton.ac.uk) for discussion prior to the commencement of research. This does not mean that you will not be able to do the research, but it will need to be considered by the School Research Ethics and Governance Committee.

Ethics forms, example consent forms/participant information sheets and supporting guidance are available on the ***Research Ethics for Projects – CEM*** area of StudentCentral.

Signed copies of this completed ethics form must be submitted with your project or dissertation. Note: the project or dissertation will not be marked if the completed checklist is not included.

PROJECT DETAILS

1. Name of researcher: George Lancaster

2. Name of supervisor: Stelios Kapetanakis

3. Title of project:

Using Machine Learning to Recognise Patterns in the Application of Optical Character Recognition

4. Outline of the research (up to 100 words):

The experimentation of machine learning methods to create a handwriting recognition web application.

5. Location of research: The University of Brighton

8. Email address: gl162@brighton.ac.uk

9. Contact address: Holly House, Horseshoe Hill, Gt Hormead, Herts, SG90NL

10. Telephone number: 07581781417

Please tick the appropriate box and answer the questions where appropriate.	Yes	No
1. Does the study involve **participants who might be considered vulnerable** due to age or to a social, psychological or medical condition? (*e.g. children, people with learning disabilities or mental health problems, but participants who may be considered vulnerable are not confined to these groups).* If yes then provide details of any such participants. See the University's 'Guidance on Good Practice in Research Ethics and Governance' for more details. …… …… Note: proposals involving vulnerable participants are often likely to require ethical approval from the Faculty of Science & Engineering Research Ethics and Governance Committee (FREGC).		X
2. Will **photographic or video recordings** of research participants be collected as part of the research? If yes then please outline consent and data protection procedures *(e.g. interviews cannot be overheard, details will not be accessible to others),* for the use of participants' images. Example consent and information forms can be found on StudentCentral and see guidance on data collection at the end of this document. …… …… If your data will not be confidential and anonymous then outline the justification for this decision here and procedures for mitigating against potential harm. …… ……		X
3. Does the study require the **co-operation of an individual to gain access** to the participants? (*e.g. a teacher at a school or a manager of sheltered housing)* If yes then describe the procedures that will be put in place to ensure safe and ethical direct involvement of human participants. Where necessary and as appropriate, include comments on obtaining informed consent, reducing harm, providing feedback, and accessing participants through an individual providing information such as a teacher/lecturer, manager, employer etc. Example consent and information forms can be found on StudentCentral. …… ……		X
4. Will the participants be asked to discuss what might be perceived as **sensitive topics** (*e.g. sexual behaviour, drug use, religious belief, detailed financial matters) or* could participants experience psychological stress, anxiety or other negative consequences (beyond what would be expected to be encountered in normal life)? If yes then describe the procedures that will be put in place to ensure safe and ethical direct involvement of human participants. Where necessary and as appropriate, include comments on obtaining informed consent, reducing harm, providing feedback. Example consent and information forms can be found on StudentCentral. …… ……		X
5. Will individual participants be involved in **repetitive/prolonged testing or vigorous physical activity, experience pain of any kind, or be exposed to dangerous situations, environments or materials** as part of the research? If yes then describe the procedures that will be put in place to ensure safe and ethical direct involvement of human participants. Where necessary and as appropriate, include comments on obtaining informed consent, reducing harm, providing feedback. Example consent and information forms can be found on StudentCentral. …… ……		X

6. Will members of the public be **indirectly involved** in the research without their knowledge at the time? (*e.g. covert observation of people in non-public places, the use of methods that will affect privacy)*. If yes then provide brief details here *(e.g. how they will be involved and, where known, the age, gender, ethnicity and location of those who will be indirectly involved).* …… …… ` Provide details of any negative impacts members of the public will be likely to face and that would not be considered minimal impacts (e.g. invasion of privacy, harm to property, being subject to what an individual perceives to be inappropriate behaviour). Describe the risks and if appropriate explain why you believe they are only minimal. …… …… Describe any procedures that will be put in place to ensure safe and ethical indirect involvement of members of the public (*e.g. providing information and feedback if requested by the public*). Examples of participation information forms can be found on StudentCentral. …… …… Describe how you will ensure data collection is confidential and anonymous (*e.g. people will not be able to be identified by photographs or notes taken by observers*), how data will be stored and who will have access to the data. If the data will not be confidential or anonymous, outline the justification for this decision here and procedures for mitigating against potential harm. …… ……		X
7. Does this research include **secondary data** that may carry personal or sensitive organisational information? *(Secondary data refers to any data you plan to use that you did not collect yourself, e.g. datasets held by organisations, patient records, confidential minutes of meetings, personal diary entrie).* If yes then provide details regarding any secondary data to be used that may carry sensitive personal or organisational information. …… …… If secondary data CEMs containing sensitive personal or organisational information are to be used, outline how such use will be ethically managed *(e.g. details such as anonymising data CEMs, ensuring protection of source agency, gaining consent of data owners, and how the data will be stored)*. See guidance on data collection at the end of this document. …… ……		X
8. Is this research likely to have significant **negative impacts on the environment**? (*For example, the release of dangerous substances or damaging intrusions into protected habitats.)* If yes then provide details of these impacts here (for example the release of dangerous substances or damaging intrusions into protected habitats) and …… …… Describe how you will mitigate against significant environmental harm and manage risks. …… ……		X

9.	Will any participants receive **financial reimbursement** for their time? (*excluding reasonable expenses to cover travel and other costs*). If yes then provide details and a short justification (e.g. amounts and form of reimbursement). …… ……		X
10.	Are there any **other ethical concerns** associated with the research that are not covered in the questions above? If yes then give details here. …… ……		X

All Undergraduate and Masters level projects or dissertations in the School of CEM must adhere to the following procedures on data storage and confidentiality.

All data should be encrypted and stored securely. Documentation should be kept in a locked cabinet or desk, and electronic data should preferably be kept on a removable disk or data stick which can be locked away, or if this is not possible on a password protected computer. Confidential and sensitive data should not be emailed unless it is encrypted or password protected since emails are centrally archived.

For Undergraduate/Masters projects, normally only the student and supervisor will have access to the data (see the University's 'Guidance on Good Practice in Research Ethics and Governance for further details).Once a mark for the project or dissertation has been published, all data must be removed from personal computers, and original questionnaires and consent forms should be destroyed unless the research is likely to be published or data re-used. If this is the case a justification for this should be included where appropriate in this form and in the relevant consent and participant information forms.

Student: Please sign below to confirm that you have completed the Ethics form and will adhere to these procedures on data storage and confidentiality.

Signed (**Student**): George Lancaster
Date: 15/11/17

Supervisor: I confirm that the research ***does/does not*** (delete as applicable) include more than a **minimum level of risk**.

Signed (**Supervisor**): Stelios Kapetanakis……
Date: ….15/11/17……..

Note: If the **supervisor judges** that there is more than the **minimum level of risk** then your supervisor will need to email this form to the CEM ethics committee (CEMethics@brighton.ac.uk) for discussion prior to the commencement of research.

Appendix H

Consent Forms

Consent to Participate in the Collection of Research Data

Title of Study

Using Machine Learning in the Application of Optical Character Recognition

Introduction

- You are being asked to contribute data to be used in a project to train and test the accuracy of various machine learning techniques.
- We ask that you read this form and ask any questions that you may have before agreeing to be in the study.
- You are free to withdraw from the study at any time.
- This study is anonymous. You are not required to give your name.

Description of Data Collection

If you agree to be in this study you will be asked to do one, or both of the following things:

- Draw **at least** 10 digits for each number in the range of 0-9.
- Draw **at least** 5 letters for each letter in the alphabet.

Consent to Participate

Gender: Male (Female)

Age: 20

I consent for the data generated to be used in this study.
I understand that this data may be made accessible to the public.

☑Yes ☐No

Consent to Participate in the Collection of Research Data

Title of Study

Using Machine Learning in the Application of Optical Character Recognition

Introduction

- You are being asked to contribute data to be used in a project to train and test the accuracy of various machine learning techniques.
- We ask that you read this form and ask any questions that you may have before agreeing to be in the study.
- You are free to withdraw from the study at any time.
- This study is anonymous. You are not required to give your name.

Description of Data Collection

If you agree to be in this study you will be asked to do one, or both of the following things:

- Draw **at least** 10 digits for each number in the range of 0-9.
- Draw **at least** 5 letters for each letter in the alphabet.

Gender: (Male) Female

Age: 22

I consent for the data generated to be used in this study.
I understand that the data may be made accessible to the public.

☑Yes ☐No

Date: _____