University of Hertfordshire
School of Computer Science

BSc Computer Science (Artificial intelligence)

Module: Final Degree Project

Creating a Neural Network based off-line Optical Character Recognition System

Luke Lloyd

Level 6

2021

0.0 Abstract

Throughout this report we look at the different methodologies and functions I used to create a working deep neural network including the ReLU, and SoftMax activation functions. We also look at the optimizers and L1 and L2 regularisation and how they work in tandem to adjust the network weights and biases to generate a high accuracy value and lower loss value. Overall though this report we look at the most common practices undertaken the formation of a deep neural network using widely recognised functions.

Contents

0.0 Abstract **2**

Contents **2**

1.0 Introduction **3**
1.1 Project Introduction 3
1.2 Problem Statement 4
1.2.1 Functional Requirements 4
1.3 Project Objectives 4

2.0 Literature Review **4**
Theme 1 5
Theme 2 5
Theme 3 5
2.1 MLPs Vs CNNs 5
2.2 NN Committees 7
Overview 7
Paper analysis 8
Conclusion 8
2.3 Feature Extraction 8
2.4 Summary 10

3.0 Legal, Social, Ethical, Professional, and Risk issues **11**
3.1 Chapter Overview 11
3.2 Legal and Social Aspects 11
3.3 Ethical Issues 11
3.4 Professional Issues 12
3.5 Project Risks 12
3.6 Chapter Summary 12

4.0 Project Design **12**
4.1 Chapter Overview 12
4.2 Neural Network 13
4.2.1 Implementation 13
Libraries: 13

Layer types: 13
Loss: 15
Accuracy: 15
Optimizer: 16
L1 and L2 regularisation: 17
4.4 Summary 17
5.0 Training and Testing 17
5.1 Overview 17
5.2 Training 17
5.3 Testing 19
5.4 Summary 19
6.0 Evaluation 19
6.1 Overview 19
6.2 Personal Evaluation 19
6.3 Research Evaluation 19
6.4 Methodology and Artefact Evaluation 20
6.4.1 Future Development 20
7.0 References 20

1.0 Introduction

1.1 Project Introduction

Throughout the process of this project I wish to create a system to identify handwritten text. This system will make use of a Convolutional neural network (CNN) and will be written in Python 3. Over the course of a number of months I researched existing systems and public projects to gather a more in depth understanding of neural networks as a whole and established an outline of the requirements needed to achieve the desired outcome of an optical character recognition system; as well as, the steps essential to fulfil these requirements.

This report details the research of different neural network types considered, specifically MLPs and CNNs, along with methods of feature extraction and the impact they have on the accuracy of the final system. The Report then goes onto look at the legal, social and ethical issues and hurdles faced in creating and using such a system in specific scenarios such as a classroom or workplace, while also considering the risks associated with such scenarios.

Finally the report describes the system in its final form, looked at in two separate sections dedicated to the architecture of the system and again the feature extraction method used. A final look at the process of training the CNN and testing the system leads into an evaluation of the project; looking into the research, methodology and artefact.

1.2 Problem Statement

Throughout this project I wish to develop a solution from an optical character recognition system that can take the input of hand written text in the form of an image and accurately predict the unicode equivalent.

1.2.1 Functional Requirements

- Easily usable for the average developer
- Universal among multiple machines and hardware setups
- Higher than 80% accuracy of classifying both digits and letters
- Save and load models

1.3 Project Objectives

I wish to gain a greater and more detailed knowledge of deep neural networks as far as the lowest level of the calculations that are required for them to be successful and reliable.

2.0 Literature Review

In this review I will be looking at several different articles related to my project. In my project project I will be looking at the use of an off-line convolutional neural networks for optical character recognition (OCR) on the case sensitive NIST (National Institute of Standards and Technology) dataset for the purpose of handwriting recognition.

Below I analyse three major themes each with three sub-categories:

Overview:

1. A description of the content covered under the topic of MLPs Vs CNNs.
2. An explanation of what a neural network committee consists of.
3. A view of the method of feature extraction and its importance in NNs.

Paper analysis:

1. Looking at the content of the papers (Ciresan *et al.*, 2011) and (Cireşan *et al.*, 2010).
2. A comparison of the implementation of NN committees as shown in (Cireşan *et al.*, 2011) and (Ciresan *et al.*, 2011).
3. An analysis of the papers (Pradeep, Srinivasan and Himavathi, 2011) and (Rajashekararadhya and Ranjan, 2009).

Conclusion:

1. My thoughts on the paper analysis and how it affects decisions about my project, and a brief comparison to more complex methods shown in (Deng, 2012).
2. I discuss the decisions I have made based on the content in (Ciresan *et al.*, 2011) in terms of my project.
3. A decision is made about my project mainly based on the paper (Pradeep, Srinivasan and Himavathi, 2011).

Theme 1

In MLPs Vs CNNs I discuss the important decision of the architecture of my neural network and deciding between the simpler multilayer perceptron and, the slightly more complex, convolutional neural network. The reason I narrowed this theme down to just these architectures is due to preliminary research that showed me both MLPs and CNNs are very accessible to learn and are very well suited to OCR.

Theme 2

Under NN Committees I discuss the possibility of using a committee of neural networks; a committee is simply the culmination of multiple neural nets that outputs are averaged to give a more accurate result.

Theme 3

Here I talk about the other main decision I have to make when it comes to creating my own neural network, that is what form or feature extraction I will be using. This is one of the most important decisions when it comes to the accuracy of a neural network because there are many different methods of feature extraction (Mori, Suen and Yamamoto, 1992) and if the one chosen is method is not implemented correctly or not well suited for the task then the accuracy level can be drastically impacted.

2.1 MLPs Vs CNNs

Overview

The theme MLPs Vs CNNs is a very important one when deciding on the type of neural net I will use in my project as these are the two main types of architectures that are commonly used in the field of handwriting recognition. Both MLPs (Multilayer Perceptrons) and CNNs (Convolutional Neural Networks) can be used for Image classification, however an MLP takes a vector as input and a CNN takes tensor as input so CNN can understand spatial relation between pixels of images better.

When discussing the use of MLPs and CNNs for handwriting recognition we have to look at how well they work when put into practice and therefore must find evidence. In the paper (Cireşan *et al.*, 2010) the simpler MLP is discussed, and it is said:

“More than a decade ago, artificial neural networks called multilayer perceptrons (MLPs; Werbos, 1974; LeCun, 1985; Rumelhart, Hinton, & Williams, 1986) were among the first classifiers tested on MNIST”.

This indicates that MLPs where the neural net of the past when it comes to this field; however, with the use of “many hidden layers, many neurons per layer, numerous deformed training images to avoid overfitting, and graphics cards” the writers were able to greatly decrease the error rate of an MLP from “a record-breaking 0.40%” to “only 0.35%” showing that the architecture is not the only significant factor when it comes to handwriting recognition.

Paper analysis

In paper (Ciresan *et al.*, 2011) the use of a CNN is chosen over the simpler MLP and also references paper (Cireşan *et al.*, 2010) stating “an error rate of 0.35% was obtained” but then goes on to say “Such an MLP has many more free parameters than a CNN” indicating this is one of the main reasons that a CNN was decided upon.

Trial	W10	W12	W14	W16	W18	W20	ORIG
1	0.49	0.39	0.40	0.40	0.39	0.36	0.52
2	0.48	0.45	0.45	0.39	0.50	0.41	0.44
3	0.59	0.51	0.41	0.41	0.38	0.43	0.40
4	0.55	0.44	0.42	0.43	0.39	0.50	0.53
5	0.51	0.39	0.48	0.40	0.36	0.29	0.46
	Committees						
	Average 0.27±0.02			Min 0.17		Max 0.37	

Figure 1

Here (*Figure 1*) average error rate of many CNNs is given as 0.27 however as can be seen the individual CNNs rarely go above 0.45 which is still a very good error rate with no optimisation. The two papers I am looking at under the theme MLPs Vs CNNs are very similar in structure and presentation. What they are looking at is first explained, then how they went about implementing the solution, and finally comparing the result they gathered to others.

It is important to keep in mind that there are other architectures of machine learning algorithms that I have not talked about here; however these two have been extensively used for the purpose of optical character recognition and are very well suited for this task as discussed in this paper (Deng, 2012).

Conclusion

From looking at the evidence gathered I think it is safe to assume I would be better off implementing a Convolutional Neural Network as the solution to my problem as it will yield a lower error rate with little optimisation and is not significantly more difficult to implement. Even

though the multilayer perceptron has been proven to successfully produce an error rate of CNNs implemented in similar scenarios, it has always required significantly more effort on behalf of the designer to achieve this and it therefore is the logical choice for me to proceed with the CNN.

2.2 NN Committees

Overview

Many different architectures of neural networks can be used as part of a committee where multiple neural nets work in construction with each other and their individual results are averaged together to give a yet more accurate result. Under this theme I look at two papers about the use of these committees; paper (Cireşan *et al.*, 2011) looks at the use of the simple MLP whereas (Ciresan *et al.*, 2011) continues to use the CNN.

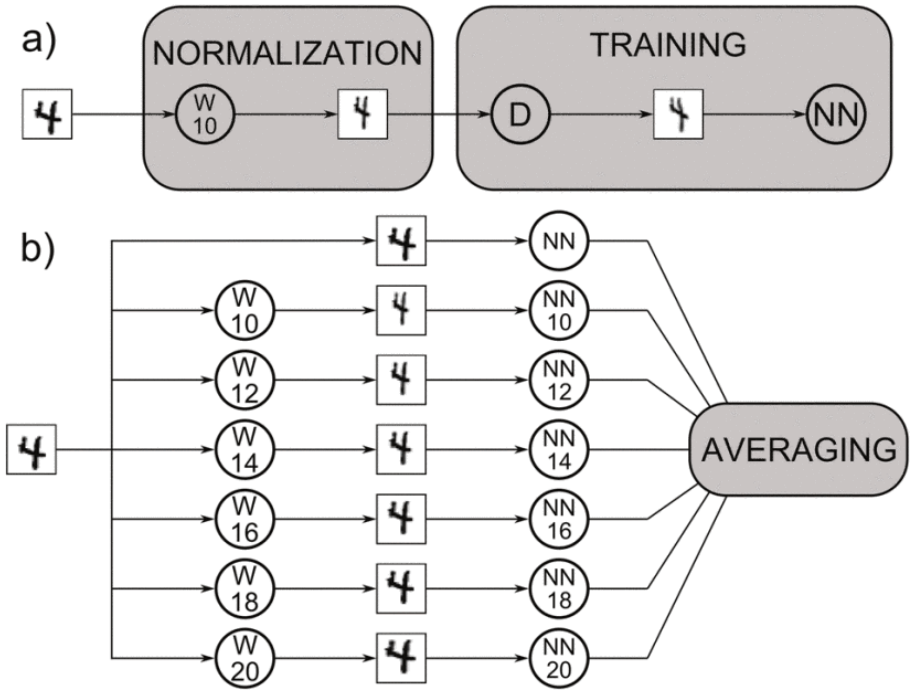

Figure 2

This image (*Figure 2*) demonstrates how a committee of either architecture is implemented. a) represents the way each network in the committee is trained individually; the training data is width normalised (W10) and is then distorted (D) for each training epoch before being fed to the neural net. b) represents the testing of the committee, if required the input characters are width normalised (W blocks) to then be processed by the corresponding neural net. the committee then averages the outputs of its NNs.

Paper analysis

When comparing these two papers about neural network committees there are very few differences as they both follow the same procedure and come to the conclusion that the use of

a committee compared to a single neural net greatly enhances the performance. However, this has certain trade-offs as these committees require a greater amount of training as every network must be trained individually on high level hardware.

Paper (Cireşan *et al.*, 2011) proves that MLPs working as a committee can bring result similar to those of CNNs it is stated that:

“On the competitive MNIST handwriting benchmark, single precision floating-point GPU-based committees of neural nets (each with a different preprocessor motivated by observed variations in aspect ratio and slant of handwritten digits) outperform all previously published methods, including complex ones involving specialised architectures, unsupervised pre-training, combinations of machine learning classifiers etc.”

In this statement it is obvious there is still potential for these simpler MLPs. However, in paper (Ciresan *et al.*, 2011) it is said that its “committee-based classifiers of isolated handwritten characters are the first on par with human performance” and therefore is quite a leap forward from the, comparatively, basic MPLs of the past.

Conclusion

In terms of my project there is little doubt in my mind that my solution would greatly benefit from a committee; however, these examples have been using the far simple MNIST dataset, apart from (Ciresan *et al.*, 2011) which reports an error rate of 21.41±0.16 when testing case sensitive letters (which I am looking at) and since time is a large factor in the creation of my solution I will opt to use a single neural network and keep a committee as an option for improvement.

2.3 Feature Extraction

Overview

Although when using the NIST dataset steps such as segmentation are not necessary, a big part of my project will be feature extraction and I will have to decide which of the many possible methods I will choose. Under this theme I will review two papers (Pradeep, Srinivasan and Himavathi, 2011) (Rajashekararadhya and Ranjan, 2009); both of these papers discuss a simple zone-based feature extraction method implemented in two different ways.

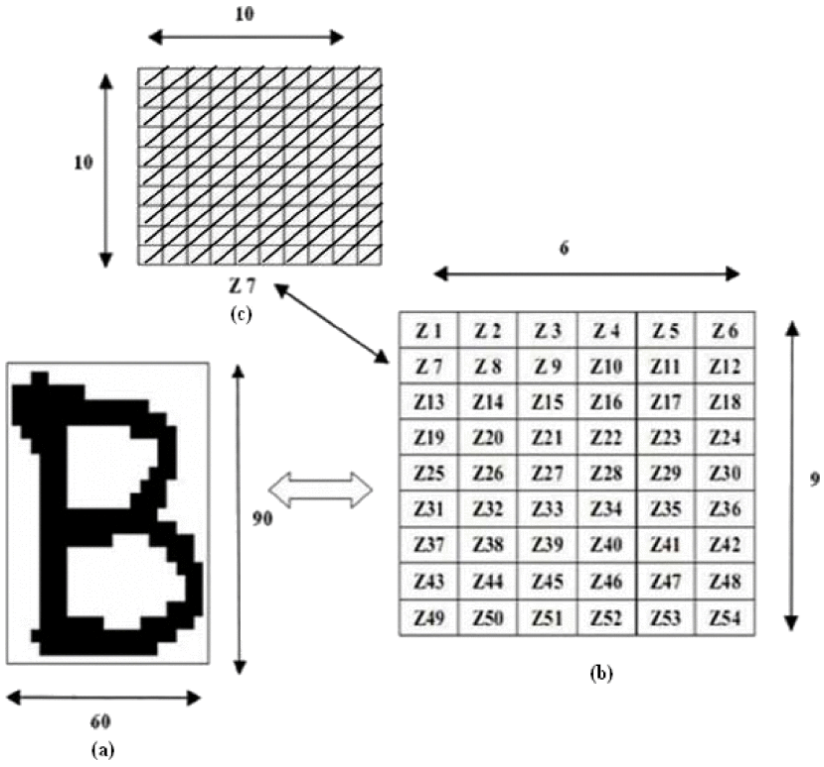

Figure 3

This image (Figure 3) is a visualisation of the diagonal feature extraction discussed in (Pradeep, Srinivasan and Himavathi, 2011) (a) shows the width and height normalised character at 60x90 pixels. This is then split into 54 separate 10x10 pixel zones depicted in (b). Each zone is then sequentially analysed along its 19 North-easterly diagonals and it is explained in the paper that:

“Each zone has 19 diagonal lines, and the foreground pixels present along each diagonal line are summed to get a single sub-feature. Thus 19 sub-features are obtained from each zone. These 19 sub-features values are averaged to form a single feature value and placed in the corresponding zone (Fig. 3(b)). This procedure is sequentially repeated for all the zones. There could be some zones whose diagonals are empty of foreground pixels. The feature values corresponding to these zones are zero. Finally, 54 features are extracted for each character. In addition, 9 and 6 features are obtained by averaging the values placed in zones rowwise and columnwise, respectively. As a result, every character is represented by 69, that is, 54 + 15 features.”

This comprehensive overview of the method was extremely useful for my understanding of the method and is something I have found lacking in other papers such as (Rajashekararadhya and Ranjan, 2009).

Paper analysis

In (Pradeep, Srinivasan and Himavathi, 2011) a diagonal zone-based method is used and is compared to the more traditional horizontal and vertical based method used in (Rajashekararadhya and Ranjan, 2009):

“The proposed recognition system performs quite well yielding higher levels of recognition accuracy compared to the systems employing the conventional horizontal and vertical methods of feature extraction.”

This statement alludes to the more traditional methods and claims that a much better accuracy rate comes as a result of this. A reported accuracy rate of 97.84% is given compared to a rate of 92.69% and 94.73% for vertical and horizontal methods respectively; however paper (Rajashekararadhya and Ranjan, 2009) reports similar accuracy rate of 97.8% by using a more intensive combination of horizontal and vertical features with a total of 1000 features compared to the much simpler 69 features gathered in (Pradeep, Srinivasan and Himavathi, 2011); therefore, the diagonal design has a drastically lower amount of data to compute, increasing its efficiency.

Conclusion

When considering the data and the methods used for basic feature extraction, I have decided to go forth with the simple low intensity diagonal based method of feature extraction proposed in (Pradeep, Srinivasan and Himavathi, 2011) as I believe it will help to provide a promising recognition rate with ease; as opposed to the classical horizontal and vertical method because it produces many more features per character and will therefore be a more intensive program.

2.4 Summary

Overall, I believe these papers have given me some very important information and insight into my project and what goes into a working solution. Due to the information I have learnt from this literature review I have decided to use a single convolutional neural network using a zone based diagonal analysis feature extraction method.

The reason for this is because I believe this will be the most practical yet effective solution that I can achieve given my limited time frame; a convolutional neural net, as opposed to an MLP, has the benefit of a higher success rate with a comparatively lower complexity threshold for a system with a similarly low error rate. This coupled with the innovative diagonal feature extraction allows for an increase in accuracy with a decrease in computational complexity; furthermore, I opted for a single neural net, instead of a committee of them as I saw no great benefit alluding to the accuracy with such an increase in difficulty and implementation time.

My biggest challenge I believe will be the construction of the CNN itself because it is notorious for having a steep learning curve, as it consists of many abstract methods and algorithms that I have not yet experienced.

3.0 Legal, Social, Ethical, Professional, and Risk issues

3.1 Chapter Overview

In this section I will be discussing the various different aspects associated with this project and how this impacts the creation process overall. This will be split
into several sub-categories:

Legal and Social: here I will discuss both the legal issues associated with the use of the final artefact along with how these and several other aspects affect its use in a social environment.

Ethical Issues: this sub-category will detail the moral hurdles that might be involved, depending on the specific use of the final project.

Project Risks: This project doesn't have any inherent risk associated with it; however, the legal and ethical problems discussed in the previous sections do pose possible risks with regards to the project's use. These risks will be discussed here.

This chapter will then be concluded with a summary of my views on the information discussed here and how I propose potential users can mitigate these issues.

3.2 Legal and Social Aspects

As far as the use of the MNIST data in the training and the testing of the initial system there is very little to worry about as the data is openly available and free for anyone to use, not to mention it is entirely open source. Due to the nature of the MNIST dataset there are no reservations with regards to its use in this project and there are no legal issues to consider.

In terms of a social environment, legal practices may have to be considered as the inputting of other people's handwriting in the use of this system should be monitored to make sure that the information gathered is not shared without the knowledge or permission of the original participant as this information could possibly be used in the making of forgeries.

3.3 Ethical Issues

In the field of ethics there is very little to be considered as the information required in the creation and subsequent use of the system is free, open source and publicly available. As far as the use of this system outside of this report, the system does not aid in any known

unethical actions apart from the possibility of improper care of provided data from third party users as discussed above.

3.4 Professional Issues

The aforementioned carelessness mainly comes into context in the systems use in such environments like classrooms, public events or businesses as in these situations the system may be used to aid in the reading of handwriting and if used on hardware with improper security the users written data could be stolen and present the ethical question of its use in these scenarios.

3.5 Project Risks

The most prominent risk of stolen information and its subsequent possible use in forgeries has already been outlined in the previous chapters. Apart from this, the only unaddressed risk, in both the creation and use of this given system stands to be the possible corruption of data vulnerable to the systems recurrent accessing of the hardware's storage. A possible version of this problem could be the use of the system on a dedicated machine or the relocation of sensitive data to a separate form of storage so as to protect from its loss.

3.6 Chapter Summary

Overall there is nigh on no issues to be considered related to the creation of this system, as well as its subsequent use in the majority of applications, from plate recognition to private use. The exception to this being considering its utilisation in situations that require the handling of others information, in which case the guidelines of GDPR should be adhered to in order to protect in the possible loss of data and said data's use in nefarious actions.

4.0 Project Design

4.1 Chapter Overview

This chapter is dedicated to detailing the process in which the artefact was created. This section has again been sub-categorized into the two main aspects of this project as discussed in the literature review:

Neural Network: here I describe the overall structure of the neural network model and how it was constructed to form a working network.

Methods Used: this section will go into depth of how the network goes about recognising and identifying different characters on a deeper level.

4.2 Neural Network

When I previously set out creating this network I intended to make it a convolutional one; however, after deciding to create the neural net from scratch instead of using API's such as Keras and TensorFlow I found the creation of a convolutional neural network challenging beyond my capabilities and I concluded I would not be able to complete this project within the given time frame. After much consideration I created a regular deep neural network using weight and bias regionalization along with batch-wise and epoch-wise loss and accuracy calculations.

The network uses the EMNIST library and the 'digits' and 'letters' datasets located within the library; these datasets are used to create two separate models that can be used in tandem to identify handwritten text.

4.2.1 Implementation

Here I will go into greater detail about the implementation of the neural network. Throughout this project I will be using python 3 and a variety of imported libraries.

Libraries:

Numpy: aids in the handling of large and multi-dimensional arrays and matrices.

EMNIST: grants access to the extended MNIST database.

Pickle: implements protocols to serialise and deserialize python objects.

Copy: provides both deep and shallow copy operations to python.

Layer types:

Layer_dense: the dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer.

As can be seen in figure 4 every dense layer is connected to the previous layer as well at the next layer in the network.

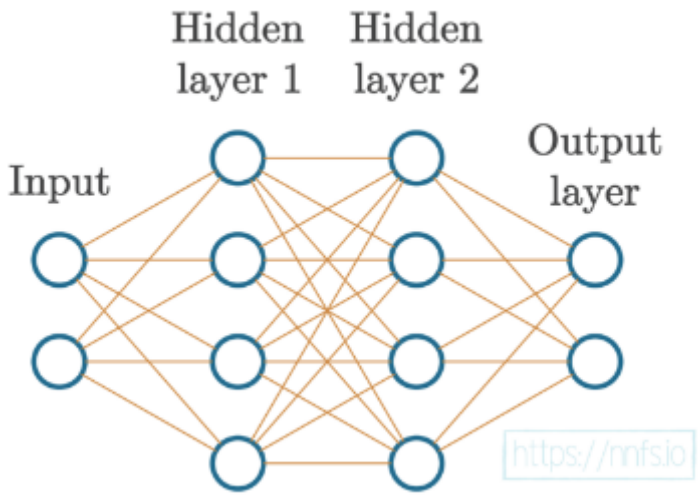

Figure 4

ReLU Activation Layer: in this project I decided to use the very common rectified linear activation function for my activation layer and in terms of how this interacts with the data it essentially only cares about data points that are above zero as it will turn any negative piece into 0.

As can be seen here in figure 5 all negative values are flattened and all other values are kept the same and this is applied to every input on that fully connected layer.

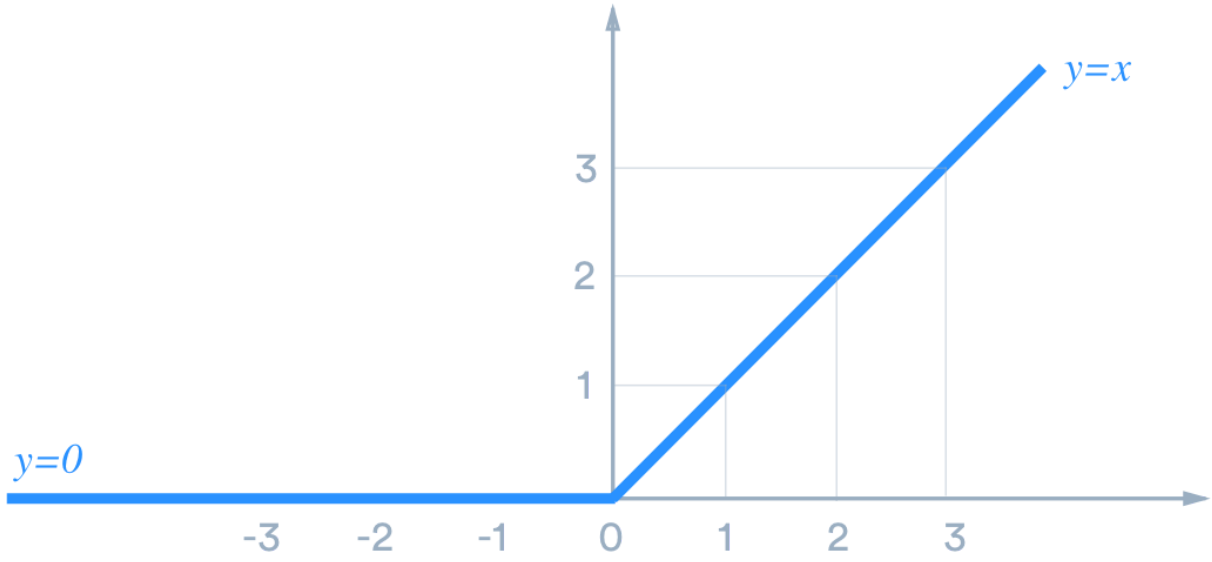

Figure 5

SoftMax Activation Layer: for the output layer I again decided to use a common activation function that is designed from classification of the data, the SoftMax function. The reason

we need a different activation function is because the rectified linear unit is unbounded, not normalised with other units,

and exclusive. “Not normalised” implies the values can be anything, an output of [12, 99, 318] is without context, and “exclusive” means each output is independent of the others. To address this lack of context, the SoftMax activation on the output data can take in non-normalized, or uncalibrated, inputs and produce a normalised distribution of probabilities for the classes.

In mathematical terms the function look like this: (Figure 6)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$

Figure 6

In simple words, it applies the standard exponential function to each element z_i of the input vector z and normalises these values by dividing by the sum of all these exponentials; this normalisation ensures that the sum of the components of the output vector σ(z) is 1.

Loss:

The model has a SoftMax activation function for the output layer, which means it’s outputting a probability distribution. Categorical cross-entropy is explicitly used to compare a “ground-truth” probability, so it makes sense to use cross-entropy here. It is also one of the most commonly used loss functions with a SoftMax activation on the output layer.

The formula for calculating the categorical cross-entropy of y (actual/desired distribution) and ŷ (predicted distribution) is: (Figure 7)

$$L_i = -\sum_j y_{i,j} log(\hat{y}_{i,j})$$

Figure 7

Where L_i denotes sample loss value, i is the i-th sample in the set, j is the label/output index, y denotes the target values, and ŷ denotes the predicted values.

Accuracy:

While loss is a useful metric for optimising a model, the metric commonly used in practice along with loss is accuracy, which describes how often the largest confidence is the correct class in terms of a fraction. Conveniently, we can reuse existing variable definitions to calculate the accuracy metric. We will use the argmax values from the softmax outputs and then compare these to the targets. To do this we combine the SoftMax and Categorical

Cross Entropy to get this: (Figure 8) , and this allows us to compare the argmax values with the target values to get our accuracy

```python
# Common accuracy class
class Accuracy:
    # Calculates an accuracy
    # given predictions and ground truth values
    def calculate(self, predictions, y):
        # Get comparison results
        comparisons = self.compare(predictions, y)
        # Calculate an accuracy
        accuracy = np.mean(comparisons)
        # Add accumulated sum of matching values and sample count
        self.accumulated_sum += np.sum(comparisons)
        self.accumulated_count += len(comparisons)
        # Return accuracy
        return accuracy
    # Calculates accumulated accuracy
    def calculate_accumulated(self):
        # Calculate an accuracy
        accuracy = self.accumulated_sum / self.accumulated_count
        # Return the data and regularization losses
        return accuracy
    # Reset variables for accumulated accuracy
    def new_pass(self):
        self.accumulated_sum = 0
        self.accumulated_count = 0

# Accuracy calculation for classification model
class Accuracy_Categorical(Accuracy):
    def __init__(self, *, binary=False):
        # Binary mode?
        self.binary = binary
    # No initialization is needed
    def init(self, y):
        pass
    # Compares predictions to the ground truth values
    def compare(self, predictions, y):
        if not self.binary and len(y.shape) == 2:
            y = np.argmax(y, axis=1)
        return predictions == y
```

Figure 8

Optimizer:

Firstly the optimizer uses a vector called the gradient, this vector made up of all the partial derivatives of a function, a partial derivative is a number calculated to represent the impact a single input has on the function.

In a neural network the optimizer is responsible for using the gradient to adjust the weights and biases of each neuron to minimise the calculated loss and maximise the accuracy of the model.

The Adam optimizer, short for Adaptive Momentum, is currently the most widely-used optimizer and is built atop RMSProp (Root Mean Square Propagation), with the momentum concept from SGD (Stochastic Gradient Descent) added back in. This means that, instead of applying current gradients, we're going to apply momentums like in the SGD optimizer with momentum, then apply a per-weight adaptive learning rate with the cache as done in RMSProp.

The explanation behind the methods used in SGD and RMSProp are outside the scope of this report. But simply put, the Adam optimizer uses the momentum mechanics as laid out by SGD as well as the per–weight adaptive learning rate as practised in RMSProp.

L1 and L2 regularisation:

Regularisation methods are those which reduce generalisation error within the model. The forms of regularisation that we'll address are L1 and L2 regularisation. L1 and L2 regularisation are used to calculate a number (called a penalty) which is added to the loss value to penalise the model for large weights and biases. Large weights might indicate that a neuron is attempting to memorise a data element; generally, it is believed that it would be better to have many neurons contributing to a model's output, rather than a select few.

L1 regularisation's penalty is the sum of all the absolute values for the weights and biases. This is a linear penalty as regularisation loss returned by this function is directly proportional to parameter values. L2 regularisation's penalty is the sum of the squared weights and biases. This non-linear approach penalises larger weights and biases more than smaller ones because of the square function used to calculate the result. In other words, L2 regularisation is commonly used as it does not affect small parameter values substantially and does not allow the model to grow weights too large by

heavily penalising relatively big values. L1 regularisation, because of its linear nature, penalises small weights more than L2 regularisation, causing the model to start being invariant to small inputs and variant only to the bigger ones. That's why L1 regularisation is rarely used alone and usually combined with L2 regularisation if it's even used at all.

4.4 Summary

Overall I have used the most commonly implemented practices and functions as they are tried and tested and widely agreed to be some of the best ways to get reasonable results in classifying datasets with the use of a deep neural network.

5.0 Training and Testing

5.1 Overview

This chapter is dedicated to the training and testing of the deep neural network that makes up the character recognition system. The role of the methods of activation, loss, optimization and accuracy will be described in the context of this project.

5.2 Training

As discussed in the previous chapter I have used multiple different functions and methods that all have a role to play in the training and testing of the network model; however, before we begin training the model there is some pre-processing of the data to make it more manageable.

Firstly we need to shuffle up the data to minimise the problem of overfitting where the model learns to simply predict purely based on the dataset it was trained on and will have poor accuracy when it comes to input data outside of the initial dataset. This isn't as simple as using a shuffle function on the training and testing data because the data and its labels will not be aligned resulting in a model that is wrong the majority of the time as it will match inputs with the wrong labels. To get around this we have to use the datasets keys, these are the same for both the data and the labels: (Figure 9)

```
#Shuffle the keys
keys = np.array(range(X.shape[0]))
np.random.shuffle(keys)

#Reorder the indexes
X = X[keys]
y = y[keys]
```

Figure 9

These newly shuffled keys are then applied to the data (X) and its labels (y) as the new indexes.

Now that we have shuffled the dataset it needs to be flattened as this neural network only works with batches of 1-dimensional vectors where as the EMNIST dataset is in the form of 28x28 2-dimensional arrays so we use the Numpy reshape operation to turn them into single dimension vectors. At the same time as doing this we can scale the data from its initial values of 0 – 255 to proportional values from -1 to 1 to reduce the stress on the system. (Figure 10)

```
#Scale  and reshape samples
X = (X.reshape(X.shape[0],-1).astype(np.float32) - 127.5) / 127.5
X_test = (X_test.reshape(X_test.shape[0],-1).astype(np.float32) - 127.5) / 127.5
```

Figure 10

For example this would take the EMNIST ‘digits’ data from the a shape of (240000, 28, 28) to (240000, 784), 28*28=784, and the 784 values contained in those 240,000 samples will only range from -1 to 1.

Now we can initialise the model and set it up for training. Training entails two main processes. We have forward propagation and backward propagation. Forward propagation is a test run through the model, in order to ascertain a loss and accuracy value of the model with its current weights and biases. This is then used in the backwards propagation by the optimiser to adjust the weights and biases, to minimise loss and maximise accuracy.

Throughout this backwards and forwards training, the L1 and L2 regularisation penalises the model for excessively large weights to counteract the generalisation error. During the forward propagation, the ReLU activation function is triggered by the input values after they

have been augmented by the weights and biases of the model's hidden layers – this is how the aforementioned weights and biases affect the model as a whole.

5.3 Testing

After each forward pass, the model is tested using the selection of test samples provided by the EMNIST dataset. The EMNIST dataset provides a balanced number of training and testing samples, meaning that there are an equal number for each separate class – for example, in the digits dataset, there are an equal number of samples for each value from zero to nine in both the training and testing sections.

5.4 Summary

To summarise, the methods and functions I have decided to use, as mentioned in the previous chapter, have been implemented in the most common and effective arrangement. I have done this because this method is widely regarded as the most effective in classifying datasets using a deep neural network.

6.0 Evaluation

6.1 Overview

The evaluation chapter will be feature my personal self evaluation as well as looking at the research undertaken prior to the project as well as the subsequent methodology that was derived from this. To finish this chapter I will take a closer look at the actual artefact and in what areas it met the proposed requirements and inversely the areas it did not. I will then propose future development that could be undertaken to take the artefact further and improve on any short comings the current system may have.

6.2 Personal Evaluation

Overall, I believe I have successfully achieved my aims of creating an optical character recognition system in a reasonable time-frame, albeit with different methods than those I set out to use. Throughout this project, I have gained a knowledge and understanding behind the inner workings of deep neural networks beyond what I originally envisioned.

6.3 Research Evaluation

Although not all of what I researched was fully utilised in the design of the neural network system, I feel that the research gathered is still relevant to this project and this report, as it

details a different form of neural network (convolutional neural network) which would benefit the goal of this system.

During my research I also looked at a form of feature extraction that I wished to integrate, however it was beyond the scope of this project, considering the time frame specified, as well as the resources at

my disposal at this current time.

6.4 Methodology and Artefact Evaluation

My methodology was sound in terms of completing the goals I set out to achieve; I created a fully functional deep neural network using this, which was able to classify hand-written digits to an accuracy of 98%. It was also able to achieve an accuracy of 83% with regards to handwritten letters.

6.4.1 Future Development

As mentioned above, I was not able to fully utilise all of my research, and successfully implement a convolutional neural network within my project. Therefore, in the future it would be valuable to create a new system that utilises these methodologies that I believe would greatly benefit in both minimising the loss and maximising the accuracy of the system as a whole.

7.0 References

Ciresan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J. 'Convolutional neural network committees for handwritten character classification'. *2011 International Conference on Document Analysis and Recognition*: IEEE, 1135-1139.

Cireşan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J. (2010) 'Deep, big, simple neural nets for handwritten digit recognition', *Neural computation,* 22(12), pp. 3207-3220.

Cireşan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J. (2011) 'Handwritten digit recognition with a committee of deep neural nets on gpus', *arXiv preprint arXiv:1103.4487*.

Deng, L. (2012) 'The mnist database of handwritten digit images for machine learning research [best of the web]', *IEEE Signal Processing Magazine,* 29(6), pp. 141-142.

Mori, S., Suen, C. Y. and Yamamoto, K. (1992) 'Historical review of OCR research and development', *Proceedings of the IEEE,* 80(7), pp. 1029-1058.

Pradeep, J., Srinivasan, E. and Himavathi, S. 'Diagonal based feature extraction for handwritten character recognition system using neural network'. *2011 3rd International Conference on Electronics Computer Technology*: IEEE, 364-368.

Rajashekararadhya, S. and Ranjan, P. V. 'Zone based feature extraction algorithm for handwritten numeral recognition of Kannada script'. *2009 IEEE International Advance Computing Conference*: IEEE, 525-528.