# A Neural Network to Process and Recognize Handwriting Using The NIST Dataset

**Kimberly Burke**

**Niall Devery**

B.Sc.(Hons) in Software Development

April 2019

**Final Year Project**

Advised by: John French, Brian McGinley

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

# Contents

# Chapter 1

# Abstract

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which are used to structure computational models. The model can take in an input image, assign importance with the use of weights and biases to various aspects/objects in the image and be able to differentiate one from the other.

The architecture of a CNN is modelled on the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. The model attempts to mimic how learning takes place in the brain.[1]

In this minor dissertation we propose to develop a CNN model in which images of a numerical and alphabetical value can be inserted and processed by our CNN model. Also for our model to successfully recognize the character and output the results.

**Authors** This is a group project being completed by Kimberly Burke and Niall Devery. We are both fourth year students in GMIT completing a level 8, B.Sc (Hons) in Software Development.

# Chapter 2

# Introduction

## 2.1 Overview

In chapter one we have our abstract introduction to the project which explains the idea of our project and a bit of information on what it is about. Chapter two discusses the introduction to the project. In this section information on the topics that we are covering are provided and what they consist of. This chapter also covers project context about our specific concept, our Project resource including our github shared space and our overall project objectives. Chapter three is our broken down literature review, where we cover the background of the technologies and processes we used. There is also some in depth information on specific technologies used. Chapter four is a continuation of our literature review but in this chapter a technology review is focused on. An in depth description of the technologies we used and how they work are described here. Chapter five contains the methodology. In the methodology we have our initial planning section, the research methodology, our project management, our project planning, our system architecture, and the project development. In chapter six the design of our system is discussed where we explain our system architecture and what elements went into the core structure of our project. Chapter seven gives an evaluation of our system, with the results of our application, the testing process and any complications and issues that we incountered during the development stage. Chapter 8 draws our project to a close with our conclusions of the entire process. This includes our finding and our future development that we would encorporate into our project to further expand and enhance our system in the future. At the very end of our paper you will find our bibliography with all of our very helpful research links attached.

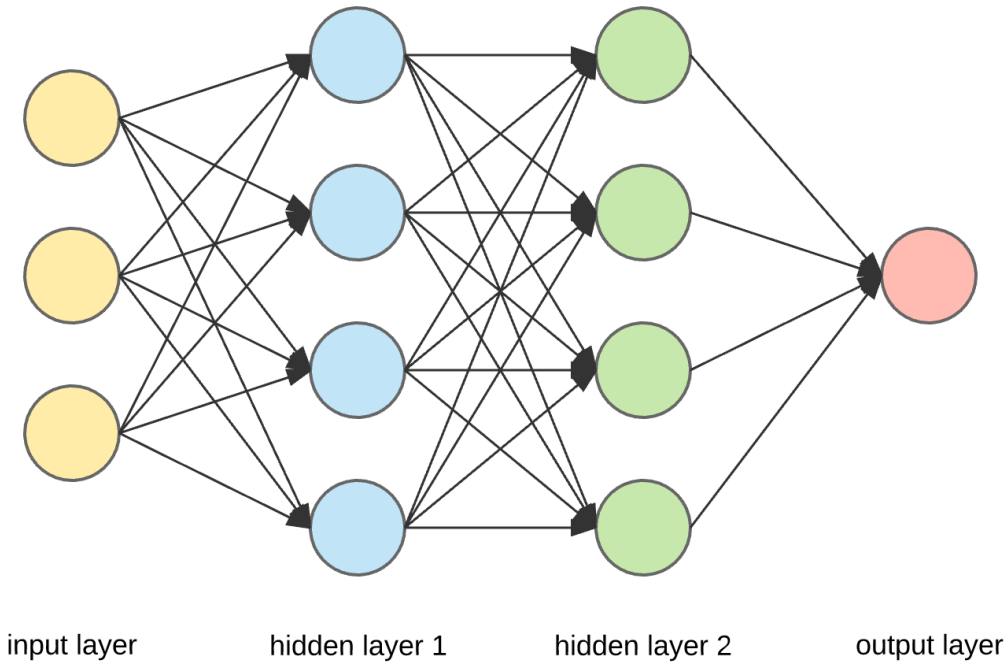



Figure 2.1: Neural network.

## 2.2 Machine learning

Machine learning is an application of artificial intelligence that allows systems to improve and learn from experience without being programmed to perform a specific task. One of the popular facets of machine learning is supervised learning, supervised learning is the task of mapping input to outputs data based on examples of training data (input) and label data (expected out). Machine learning algorithms are used commonly in email filtering and computer vision. A supervised learning algorithm produces an inferred function by analyzing the training data given to it which can be used to predict an output and map label data to the unseen samples given (testing data).

## 2.3 Neural networks

An artificial neural network is comprised of a series of layers each containing neurons, there is typically 1 input layer, 1 output layer and any number of given hidden layers (see *Figure 2.1*) that can all be connected.

Each of these layers is comprised of neurons or nodes for example the input layer of a neural network is made to recognise the MNIST data set which

is comprised 70,000 single digit numerical characters from 0-9 would consist of 784 neurons (28 X 28 pixels), one neuron for each pixel and the output layer would consist of 10 neurons, one for each classification (numbers from 0 - 9). Each one of these neurons contains a numerical value or "activation" in the case if MNIST these values are between 0 and 1. Each one of these activation's determine the activation's in the next layer. In the second layer or first hidden layer the activation for that neuron is calculated by determining the activation of the previous layers neurons and assigning "weights" or parameters to the connections between the layers. How the neural network learns is by adjusting these weights as the training data is input into the model to recognise that there are certain patterns in the data for example an 8 is comprised of pixels that form one round shape on another. A "bias" is a number associated with each neuron in all the hidden layers and outputs layers when calculating the activation of neurons when moving between layers. This bias determines how high the activation of a particular neuron must be in order for the neuron to be meaningfully active. This process is repeated until the output layer is reached in which the neurons in that layer have an activation that allows the neural network to make and reasonable prediction of what the data maybe representative of. Because the input layer is not connected to any previous layer there are no biases associated with these neurons.

When the neural network reaches the final layer the activation of these nodes are between 0 and 1 and the neural network uses the softmax to determine what node is the highest value and then takes this as the networks prediction.

The measure of a neural networks inferred function is not necessarily calculated by the overall accuracy of the model but rather the models ability to mitigate loss.

In order to solve a problem using supervised learning certain steps must be followed.

1. The type of training data must be determined. The training must be consistent throughout.

2. Gather the training set. The training set must be representative of real world data. In the case of the NIST data set it was taken from 3600 individual writers.

3. Determine the specific features that the function will learn. With a given input the function needs to infer specific features about the data in order to perform.

4. Determine what type of function corresponds to the learning algorithm.

5. Complete the design. Because there are no specifically set algorithms for any given task in machine learning it is up to the developer to determine the best structure of the neural network model by running the learning algorithm of the training set.

6. Test and evaluate results. Once to the training has taken place and the function has inferred certain features about the data, it must be tested using unseen data separate from the training set.

## 2.4 Convolutional neural networks

Feed-forward neural network in artificial neural networks is usually known as Deep Neural Networks (DNNs) this means that the neurons or "nodes" do not form a loop or a cycle within the neural network model. Convolutional Neural Networks (CNNs) are one kind of feed-forward neural network.

CNN is an decisive recognition algorithm. It is vastly used in pattern recognition and image processing. It has many features such as simple structure, less training parameters and adaptability. Its weight shared network structure make it more similar to biological neural networks. It reduces the complexity of the network model and the number of weights. In particular multi-dimensional input vector image can directly enter the network, which avoids the complexity of data reconstruction in feature extraction and classification process.[2]

A convolutional neural network usually consists of two main layers, one that handles an input and one that handles an output. It also contains many hidden layers aswell. Hidden layers in the CNN usually comprise of convolutional layers and RELU layer such as pooling layers, activation function, and normalization layers.

The objective of the Convolution layers is to appeal to the higher level features such as edges, from the input image. Usually, the first layer's job is to deal with the low-Level features such as edges, color, gradient orientation, etc. When layers are added, the architecture of the model adapts to both the low and high level features, which provides the system with a greater knowledge of the images from the data set.[1] After each convolutional layer, an activation layer is usually applied straight afterward. A convolutional layer is typically followed by a pooling layer. The most popular pooling layer is max pooling. What max pooling does is takes the output from a convolutional layer and cycles through it in blocks of neurons, the size of the blocks is set by the filter size. As seen in *Figure 2.2* the filer size is 2X2 when

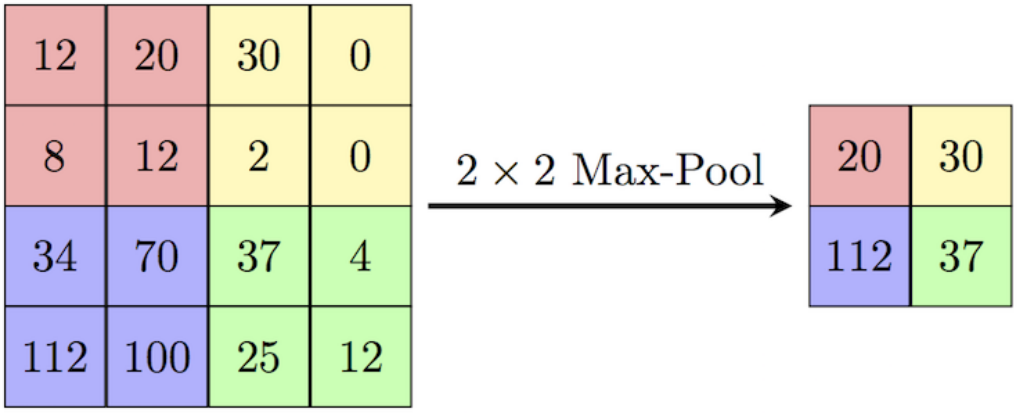



Figure 2.2: Max Pooling

the operation is occuring the max value from the grid is taken and outputted hence the name "Max Pooling". This compresses the image to extract smaller defining features.

The ReLU layer also known as the Rectified Linear Units layer of the model increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer. The ReLU layer uses the function f(x) = max(0, x). to all of the values in the input volume.

This layer basically changes all the negative activation's to 0. This layers main function is to increases the nonlinear properties of the model and the overall network without causing changes or affecting the other layers in the model. [3]

## 2.5 Context

We plan to build a CNN model in which images can be inputted, analyzed and produce a result. We are using the NIST data set which contains 810,000 images of which we will use a portion of. This data set contains handwritten characters from 3600 writers, of numbers and letters, uppercase and lowercase included, that will be hosted on a web application programming interface (API). Our web API will be executed using a combination of Python and Flask, our CNN model will be integrated into our Flask system.

## 2.6 Project Resource

https://github.com/NiallD565/DigitRecognition

## 2.7 Objectives

- To investigate how machine learning techniques may be applied to recognizing handwritten characters.
- To analyze and interpret the NIST data set.
- To determine the most accurate deep learning algorithm for the task by testing.
- To determine the best technologies to use for this task.
- To build a fully functional CNN model.
- To analyze the prediction results and compare the accuracy.
- To develop a web API that will host the trained model allows users to interact with it.
- To have the model be retrained once a certain amount of images have been uploaded to the web service.

The success of the project will be measured in four key ways:

1. The accuracy of the developed/trained CNN on the test set as a subset of the original data set.
2. The web APIs ability to take incoming information and store it appropriately.
3. The accuracy of the CNN model on input given by the users through the web API.
4. The models ability to retrain using the original training data in conjunction with the new samples taken from the users.

# Chapter 3

# Background

### 3.0.1 The background of Machine Learning and Artificial Neural Networks

In 1943 the first artificial neural network was created by a neurophysiologist named Warren McCulloch and a mathematician named Walter Pitts. They decided to create a model based upon neurons using an electrical circuit. In 1950, Alan Turing created the Turing Test. The Turing Test is a simple concept, a computer attempts to convince a human that it is also a human. The first artificial neural network was created in 1958 by Frank Rosenblatt, it was designed with the goal of shape and pattern recognition. In the next year Bernard Widrow and Marcian Hoff created two neural networks the first of them was capable of recognising binary patterns and it was named ADELINE,the second was name MADELINE it was capable of eliminating echo on phone lines and is still widely used today. Machine learning then had a slow period in which little break through were made until Japan started to fund research into advanced neural networks which prompted the US to also fund the area. There were some breakthroughs leading up to the 1990s but in 1997 the world chess champion was defeated by the IBM computer Deep Blue which was a chess playing computer. By the 21st century machine learning was becoming well known for its increasing calculation potential.

### 3.0.2 The background of Python

Python was first created in the late 1980's and began to be used in December 1989 by Guido van Rossum at CWI in the Netherlands. It was made as a sort of replacement for the ABC language. Python's principal author is Van Rossum and he was dedicated at every step to the direction in which the language took. The next full version of Python was Python 2.0, this version

was not released until 16 October 2000. [4]

### 3.0.3 Image classification

In the last decade the amount of research in image classification and recognition has drastically increased, this is partially due to the increase in computational power. The the trend of increasing computational power has been coined in 1970 as "Moore's law" which states that processor speeds, or overall processing power for computers will double every two years. How this relates to image classification is that the increase has allowed for wider research into the more computational heavy tasks. [5] States that

> "Since 2010 deep neural networks have greatly profited from Graphics Processing Units (GPU)."

The results found in [5] saw a state of the art results for the time when preformed on the MNIST handwritten data set. These results were obtained by using CNNs, which in the past would have taken months to train on a large data set, but due to the composition of CNNs along with the GPUs the computational overhead is mitigated. It is further stated that the biggest architectural difference between modern CNN implementation and CNNs developed in 1998 in the is the use of max-pooling layers instead of subsampling layers.

Max-pooling layers are used in conjunction with convolutional layers is that convolutional layers extract features from the input while max-pooling is used to identify the most important features by reducing its dimensionality to for assumptions to be made. Although this approach has been proven to be accurate it raises the issue of image distortion. If an image is of low quality or has motion blur the process of reducing the dimensionality may be to its detriment, [6] assesses the affects of image distortion in a number of ways. Under the assumption that input images to a CNN are of high quality the CNN would naturally have the ability to classify and recognise images of the same quality and the reverse with regards to lower quality images could be true.

The paper [6] examines the effects of Gaussian noise, motion blur, and reduced contrast on the original images in the MNIST data set and find minimal recognizable features in the images. It also finds that all of the networks tested were very sensitive to blur, so sensitive in fact that even when a moderate blur level was applied to the images the accuracy of the networks would decrease dramatically. This can possibly be due the the blue distorting edges in the images in addition to removing textures. The paper concludes that all of the networks tested were susceptible to blur and noise

distortions, and are resistant to compression artifacts and contrast. Further conclusions were that training with low quality images should improve testing on low quality images but may decrease performance on high quality images.

Since the development of GPUs and the study of the image recognition have seen an increase in recent years, there is discussion regarding types of classifiers, in comparing them based on not only accuracy but also rejection, training time, recognition time and memory requirements. [7] compares classifiers that are trained to recognise a subset of the NIST data set of which MNIST is derived from. A baseline linear classifier takes the input and assigns an activation to neuron in the layers, each of these neurons contribute to the weighted sum and is then calculated. The baseline linear classifier is known as the basis of comparison for more complex classifiers, as it requires different combinations of activation function and layer sizes.

### 3.0.4 Python and Flask

How Python and Flask Supports Web Development Efficiently:

Python is a general purpose, high level programming language that focuses on the code readability, for web development lines of code will be fewer than other languages. It is possible for Python because of large standard libraries which make the web development code simple and short. These libraries have pre-coded functions provided by Python community which can be easily downloaded and can be used as per the development needs. Initially Python was designed for web servers to deal with the incoming traffic on the server.

Flask is a micro framework of Python which provides the basic functionality of web framework and allows more plug-ins to be added so the functionality and feature set can be extended to a new level. Flask is called as micro framework of Python because it makes the core functionality simple but extensible in terms of development. It can also be used to save time building web applications.

Flask uses Jinja Template Engine and the Werkzeug WSGI Toolkit. Flask structure is categories into two parts “Static files and Template files”, template file have all the Jinja templates including Html pages, where as static file have all static codes needed for website such as CSS code, JavaScript code and Image files.[8]

# Chapter 4

# Technology Review

In this section we will be discussing the merits and possible drawbacks of the technologies used in the development of this project.

### 4.0.1 Visual Studio


Visual studio is an integrated development environment (IDE) made by Microsoft. It is used to create and develop, applications, websites web applications, web services and mobile apps under the .Net Framework. Visual studio is typically used to develop programs in C sharp but also supports 35 other languages to varying degrees, these languages include C, C++, C++/CLI, XML, Visual Basic, Python, .NET, JavaScript, TypeScript, HTML and CSS to name a few although some languages such as Ruby and Node.js can are available through plug-ins.

One of the features of Visual studio is the IntelliSense which allows the IDE to predict what the developer may be typing and make suggestions this

cuts down on developing time greatly as apposed to manually typing code where mistakes can be made.

### 4.0.2 Visual Studio Code


Visual studio code is an open-source code editor with built-in support for Javascript and Typescript. The key difference between Visual Studio and Visual Studio Code is that Visual studio is an IDE, while Visual Studio Code is more lightweight and has a much smaller installation size.

### 4.0.3 Anaconda


Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing such as data science, machine learning applications, large-scale data processing and predictive analytics. Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently. It has gained popularity due to it simplicity to use as well as the ease to install packages through the terminal using commands such as *conda install* and *pip install* which install packages from the Anaconda repository.

### 4.0.4 Jupyter Notebook


The Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, statistical modeling, graphical data representation and machine learning. Jupyter Notebook is included in the Anaconda packages and is freely available. Due to cells with the notebooks it is quite popular for presentation of somewhat complex code which would otherwise seem unappealing as just chunks of text on a screen.

### 4.0.5 Tensorflow

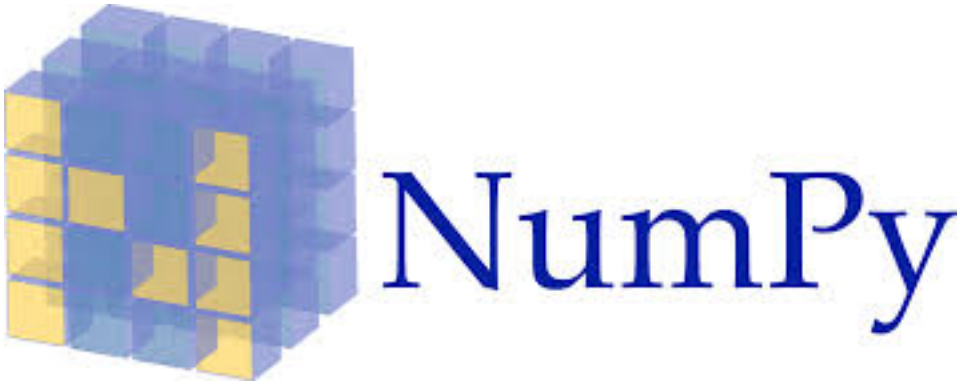

Tensorflow is an open source framework that is designed to allow for high performance numerical computation for machine learning. Though it is widely used in conjunction with Python it can also be used with C and C++. Tensorflow creates dataflow graphs, in simple terms a dataflow graph describes an input, some form of operation and an output. In Tensorflow each node

represents a mathematical function to be carried out. Tesnorflow is used to create large scale neural networks with an multitude of layers. It is commonly used for deep learning or machine learning problems such as Classification, Perception and Prediction.

### 4.0.6 Keras


Keras[9] is an open-source neural-network library written in Python for deep learning. It is capable of running on top of TensorFlow. Keras is a high level API, is considered to be more user friendly and can be run on top of Tensorflow. Keras allows the user to quickly adapt a neural network to perform a task due to its simplicity.

### 4.0.7 Numpy




Numpy[10] is a Python library that allows large, multi-dimensional arrays and matrices to be manipulated and also contains high-level mathematical functions to manipulate these functions to operate these arrays and matrices. The Python programming language was not initially designed for numerical computing so Numpy was developed. At its core it has n-dimensionality

arrays or ”ndarrys” and these can be used by converting Pythons traditional list structures to the ndarrays to perform mathmatical function. It also allows for image preprocessing in the form of arrays to format them correctly.

### 4.0.8 HTML


HTML is an acronym for HyperText Markup Language. It is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people and a powerful creative tool for website design, creating a user friendly structure. We used HTML to create the structure for our front end web page in our web application. [11]

### 4.0.9 CSS


CSS is an acronym for Cascading Style Sheet. It is a fundamental web language for describing the presentation of web pages. It defines how HTML elements are to be displayed on the screen. CSS is responsible for the look and the style of the web page. CSS rules are often reused across multiple parts of a page and across multiple pages throughout a web site to reduce repetition and to provide a consistent look and feel. [12]

### 4.0.10 Javascript


JavaScript is an object oriented scripting language, it simply tells the web browser what to do. The web browser interprets the script and does all the work. It also provides the chance to manipulate aspects of the web browser such as its height, width and position. It is commonly used to used to create interactive effects within web browsers. [13]

### 4.0.11 Github


Github is a web-based hosting service for version control that allows users to upload or ”commit/push” their computer code. It offers a number of services to users including reverting code to a previous version, task management and bug tracking. It also offers enterprise, pro and team accounts although free accounts are commonly used to host open-source software projects.

### 4.0.12 Python


Python is an object orientated, high level programming language. It is powerful and fast, and operates well with other technologies and languages. It is easy to learn and has many open source resources available for users. Python source files use the ".py" extension and are called "modules."

In Python, there are no type declaration of variables, parameters, functions, or methods in the source code. This makes the code short and flexible, and you lose the compiler-time type checking of the source code. Python tracks the types of all values at run time and flags code that does not make sense as it runs making it quick and efficient.

Advantages: Python has an extensive library and packages built in to the downloader package, this makes adding packages to a system very accessible and easy for the user. It also uses simple syntax making it user friendly and easy to learn. [8]

### 4.0.13 FLASK


Flask is a lightweight web application framework written in python and baseband on the WSGI toolkit and jinja2 template engine. Flask takes the flexible python programming language and provides a simple template for web development. Once imported into python, Flask can be used to save time building web applications. It keeps the core simple but extensible. It has no database abstraction layer, form validation, or any other components. Flask supports extensions. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and more. [8]

### 4.0.14 MySQL

MySQL is an open-source relational database management system (RDBMS).Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. MySQL is written in C and C++. It works on many systems such as Linux, macOS, Microsoft Windows and more. MySQL has gained popularity not only because it is open-source but also due to the reliable performance, simplicity and in depth documentation[14]. In this project MySQL was used to store and organise the database containing the data set as well as the user input.

### 4.0.15 Machine Learning in Python: Benefits

Python is the language of choice when it comes to machine learning and Neural Networks. A reason for this is the simplicity of the language. It is a good language as it easy to understand and to program with. It is highly popular, especially among experts from other academic backgrounds who have little programming experience but would like to take advantage of the advanced computational processes.

Another good benefit of using Python for machine learning is the strong open source community. There is such a great support and amount of resources available. Partially because of the simplicity referred to earlier, this has lead to a strong community of academics formed around the open source community. This allows libraries to evolve and improve at a rapid pace. Many Python libraries utilize cutting edge techniques and strong documentation pages. Many Python libraries also use nonrestrictive licenses, which also prompts businesses to use them and contribute back to the community.

Due to the large community and popularity of the language, Python has sophisticated scientific libraries. As many experts in the scientific fields (wishing to reach the largest possible audience) focus exclusively on developing libraries in Python. This has lead to a mature, production tested ecosystem for developing applications in Python. [15]

### 4.0.16 Technological Advantages of Python and Flask

Python and Flask go hand in hand to make a desirable environment for producing a system suitable for machine learning and integrated Neural Network models. Python and flask provide extensibility. Extensibility in web development is a principle rule designed as a system's ability to have new functionality extended, in which the system's internal structure and data flow are minimally or not affected. This means that recompiling or changing the original source code is unnecessary when changing a system's behavior.

This is a great advantage as systems are constantly modified for new features and added functionalities.

Another advantage is robustness. Which is the ability of system to cope with errors during execution. Robustness is also used as the ability of an algorithm to continue operating despite abnormalities in input, calculations, etc. Robustness can encompass many areas of web development.

Python and Flask are an open source languages in which the source code is available to the general public for use and/or modification from its original design. Open-source code is typically a collaborative effort where other developers can improve upon the source code and share the changes within the community so that other members can help improve it further.[8]

## 4.1 Technologies Used

- Anaconda
- Jupyter notebook
- Python flask
- Visual studio 2017
- Visual studio code
- Keras
- Tensorflow
- Python 3
- MySQL
- Numpy
- HTML
- CSS
- Javascript
- Github
- Overleaf

# Chapter 5

# Methodology

In this section, project planning, organisation, decisions made throughout the projects life cycle will be discussed. The differences between the initial plan and final implementation will be detailed.

## 5.1 Initial Planning

Once the specification had been given the group met and decided what technologies would the members be interested in developing as the applied project. Research was carried out into multiple areas and a consensus would met that the group would like to develop some sort of machine learning project. More research was carried out into the areas in which machine learning may be applied such as facial recognition, supermarket stock prediction and image classification. While all of these concepts interested members and could be considered within the scope of the applied project the group decided that image classification would present challenges and learning experiences. The group also decided that a form of web API would be incorporated to the project to demonstrate the abilities of the members. A meeting with supervisors was organised and held, ideas were presented and discussed in terms of plausibility and scope. The group was given advice and resources to conduct further research. Research was carried out into image classification and digit recognition, more specifically the MNIST data set.

## 5.2 Research Methodology

The members of the group had varying levels of understanding in terms of machine learning and digit recognition so some major preliminary research

was into the concepts was required to bring all members to the same general understanding of the concepts.

**Machine Learning:** Due to the teams somewhat previous experience in the area of machine learning, in depth research was needed in order to fully grasp the complexity the concept in its relation to image classification and to develop the groups knowledge in the possible types of classifiers that could be used. It was decided that *Python 3* would be used to the develop the neural network. While team members had some experience in building simple neural networks an increase in understanding and skill to develop a more comprehensive neural network would be required. Once it was decided that *Python 3* would be used the group needed to further investigate *Numpy* [10] as it would be used greatly in the python scripts in the project. With regards to attempting to classify images one member suggested tensorflow as a framework due to the groups previous experience with it in conjunction with *Python 3*.

**data set:** After considerable research into possible applications of machine learning and neural networks in image recognition and classification there was considerable development in neural networks pertaining to the MNIST data set discussing the different types of classifiers and their accuracy's. The group came across the NIST data which was developed in 1994, and decided that this data set would be sufficient in size and representative of real world data due to the wide variety of writers the data set was taken from ranging from high school students to census takers. Due to the incredible size of the original data set of 810,000 images (128X128 pixels per image) the group decided under the advice of the supervisors that taking a subset of this was appropriate as the complete data set would be computationally heavy and require a large amount of memory.

**Neural Networks:** As part of the research for this project a more in depth look at the types of neural networks was warranted as all members of the group had only developed basic models as previous experience. After reading of the accuracy and relatively small cost of running a CNN as apposed to a typical feed-forward deep neural network it was decided that this would be an efficient and appropriate approach to developing a neural capable of classifying the data set.

## 5.3 Project Management

As a part of our project management, we began our planning and time line for this project by creating a Gant chart. From this chart we added in our time line, our milestones and our goals for collectively completing this project

| | Task Mode | Task Name | Duration | Start | Finish | Predecessors | Resource Names |
|---|---|---|---|---|---|---|---|
| 1 | | Project planning and research | 4.2 wks | Mon 15/10/18 | Mon 12/11/18 | | Kim,Niall |
| 2 | | Acquire the NIST dataset merge files | 10 days | Mon 19/11/18 | Fri 30/11/18 | | Niall |
| 3 | | Re-size and re-structure the merged file within the NIST dataset | 2 wks | Mon 03/12/18 | Fri 14/12/18 | 2 | Kim,Niall, NIST[1] |
| 4 | | Create a model for the neural network | 4 days | Mon 17/12/18 | Thu 20/12/18 | 3 | Kim,Niall |
| 5 | | Create a training and testing set from the new dataset build | 3 wks | Mon 17/12/18 | Fri 04/01/19 | 3 | Kim,Niall, NIST[1] |
| 6 | | Train the neural network | 4 wks | Mon 07/01/19 | Fri 01/02/19 | 5,4 | Kim,Niall, Restuctured |
| 7 | | Document and Test | 2 wks | Mon 04/02/1 | Fri 15/02/19 | 6 | Kim,Niall |
| 8 | | Testing and finalising documentation | 2 wks | Mon 18/02/19 | Fri 01/03/19 | 7 | Kim,Niall |
| 9 | | Build a neural network proto type using MNIST | 7 days | Tue 20/11/18 | Wed 28/11/18 | | Kim,MNIST[1], Niall |
| 10 | | Testing for MNIST | 5 days | Thu 29/11/1 | Wed 05/12/1 | 9 | Kim,MNIST[1],N |
| 11 | | Creating a server through VM | 2 days | Wed 12/12/18 | Thu 13/12/18 | | Google Platform |
| 12 | | Creating a GUI for the server | 7 days | Fri 14/12/18 | Mon 24/12/18 | 11 | Google Platform |
| 13 | | Create Python Script for pre-processing | 14 days | Mon 17/12/18 | Thu 03/01/19 | 3 | Kim,Niall, Restuctured |

Figure 5.1: Project Plan

as a group within our timescale. For ensuring communication within the group we used collaborative software, where a shared space was provided so both members could contribute. We used GitHub for sharing and uploading our code from the project. We used Overleaf for a shared document write up for our dissertation. We attended weekly meetings in the collage with our project advisers for project updates and progression reports. We kept communication open between the group with regular messaging for updates and any issues that arose.

## 5.4 Project Planning

The initial plan for the project was to develop a simple yet functional web API that functions by using a saved model of a neural network to classify user input in the form of images of alphanumeric characters.

**data set:** Initially the group had planned to use a portion of the merged data set from NIST which takes both upper and lower case letters that are essentially the same and puts them into the same classifier e.g. C and c would be classified together bringing the number of classifications from 62 to 48. A subset of the merged data set was created by taking approximately 5,000 from each character to create a new data set of 240,000 images. This

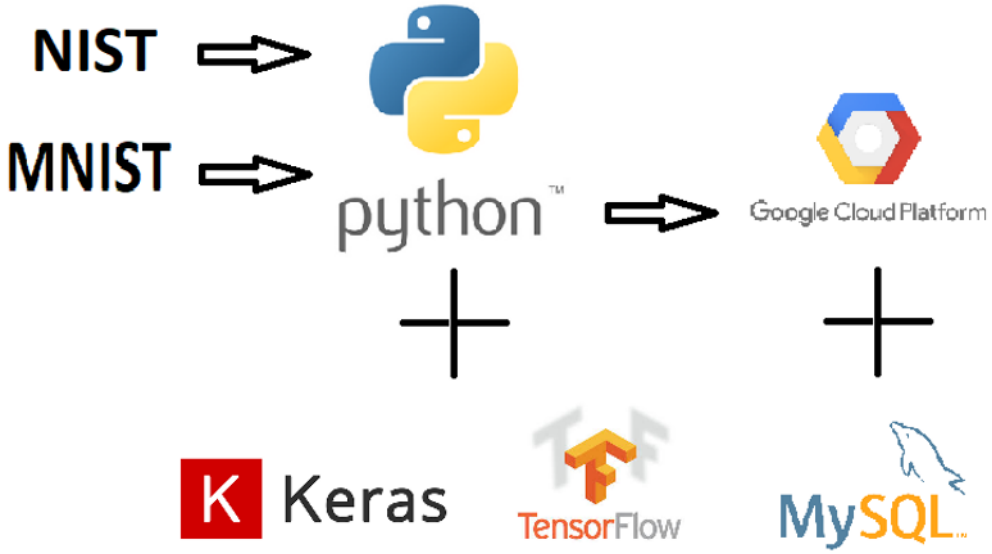



Figure 5.2: Initial System Architecture

had later changed to taking samples only from the numbers 0 - 9 and A, B, C, D, E in uppercase as the data set was far too large to allow for training due to hardware constraints.

**Neural Network:** The initial plan was to train the neural network using a Virtual Machine on google cloud where the group could access the shared machine. The rational behind this choice was to access additional computational power as Google Cloud Platform has options to use severs that have more Central Processing Units (CPUs) and as a result the training time for such a large data set would be reduced to a degree.

**Web API:** We began our research with looking for a suitable API that would cohesively work with a neural network model. Upon investigation we found that the flask micro-framework specialized in model integration and hosting a web application which was the exact system that we were searching for.

### 5.4.1 Initial System Architecture

Based upon research carried out the architecture depicted in *Figure 5.1* was agreed on as a preliminary architecture as at this point nothing was concrete and was subject to change due to possible constraints or lack of foresight.

## 5.5 Project Development

The group divided the project into the front end and back end Kimberly was tasked with the front end web API while Niall was was tasked with the back end neural network. Project meetings were held every week with alternating supervisors, the purpose of the meeting was to update the supervisors on progress and have possible queries answered. If the group had decided that not a substantial amount of work had been completed they would notify the supervisors and attempt to reschedule.

### 5.5.1 Incremental Development

Niall developed a standard feed-forward neural network to test on MNIST as proof of concept and also to gain a better understanding the process of training a neural network. As shown in *Figure 5.4* the number of correct predictions above 96 percent. This was a reasonable starting point the next step was was to develop a CNN capable of classifying the MNIST data set in less training time, number of epochs and improve overall accuracy.

Niall then developed a CNN based off of previous models discovered in research.

As seen in *Figure: 5.4* the composition of a CNN is much smaller and does not require as many parameters and as such will reduce training time. As seen in *Figure 5.6* the training times have decreased to less than a minute per epoch and the accuracy increases at a faster rate. An evaluation on both neural networks was carried out using the same test set from MNIST, as seen in *Figure 5.4* the accuracy of the basic feed-forward neural network is substantially lower when compared to the CNNs accuracy of approximately 98 percent as seen in *Figure 5.7*.

Kimberly researched possible solutions for developing a web API initially we had just planned to use a standard format for it but after discussing the findings and flask being recommended to us we decided to use it because it relationship with python. It also boasted great functionality for trained model integration which was a key feature that our project needed.

Due to the further research carried out into the NIST data set the group had found that the training set in the original data was not entirely representative of real world data. The test set was comprised of samples only taken from high school student where as the training set was comprised of other samples. In order to get a generalisation of the data the group had came to the conclusion that creating a subset of the original data set would be in the best interest of the project as it would yield more accurate representation

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 784)               615440
_________________________________________________________________
dense_2 (Dense)              (None, 455)               357175
_________________________________________________________________
dense_3 (Dense)              (None, 250)               114000
_________________________________________________________________
dense_4 (Dense)              (None, 170)               42670
_________________________________________________________________
dense_5 (Dense)              (None, 120)               20520
_________________________________________________________________
dense_6 (Dense)              (None, 50)                6050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                510
=================================================================
Total params: 1,156,365
Trainable params: 1,156,365
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.3: Basic Feed-Forwards Neural Network

```
Epoch 1/10
60000/60000 [==============================] - 32s 528us/step - loss: 0.7871 - acc: 0.7400
Epoch 2/10
60000/60000 [==============================] - 50s 827us/step - loss: 0.3150 - acc: 0.9067
Epoch 3/10
60000/60000 [==============================] - 55s 910us/step - loss: 0.2457 - acc: 0.9282
Epoch 4/10
60000/60000 [==============================] - 52s 866us/step - loss: 0.2035 - acc: 0.9394
Epoch 5/10
60000/60000 [==============================] - 53s 877us/step - loss: 0.1636 - acc: 0.9513
Epoch 6/10
60000/60000 [==============================] - 52s 874us/step - loss: 0.1562 - acc: 0.9540
Epoch 7/10
60000/60000 [==============================] - 53s 880us/step - loss: 0.1331 - acc: 0.9605
Epoch 8/10
60000/60000 [==============================] - 53s 876us/step - loss: 0.1254 - acc: 0.9628
Epoch 9/10
60000/60000 [==============================] - 54s 894us/step - loss: 0.1193 - acc: 0.9645
Epoch 10/10
60000/60000 [==============================] - 53s 891us/step - loss: 0.1080 - acc: 0.9676
```

Figure 5.4: Accuracy of training

```
Number of correct predictions out of 10,000:
9637
```

Figure 5.5: Basic NN result

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 28)        280
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 28)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4732)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               605824
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 607,394
Trainable params: 607,394
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.6: Convolutional Neural Network Structure

```
Epoch 1/7
60000/60000 [==============================] - 57s 955us/step - loss: 0.2076 - acc: 0.93800s
Epoch 2/7
60000/60000 [==============================] - 57s 956us/step - loss: 0.0853 - acc: 0.9736
Epoch 3/7
60000/60000 [==============================] - 58s 960us/step - loss: 0.0593 - acc: 0.9817
Epoch 4/7
60000/60000 [==============================] - 58s 960us/step - loss: 0.0461 - acc: 0.98500s
Epoch 5/7
60000/60000 [==============================] - 62s 1ms/step - loss: 0.0373 - acc: 0.9879
Epoch 6/7
60000/60000 [==============================] - 58s 969us/step - loss: 0.0301 - acc: 0.9900
Epoch 7/7
60000/60000 [==============================] - 59s 991us/step - loss: 0.0260 - acc: 0.9912
```

Figure 5.7: Convolutional Neural Network Training

```
Error percentage: 1.61%
```

Figure 5.8: Convolutional Result

of the neural networks predictions by taking random samples, therefore the group had selected images from the merged data set at random. Because of the suspicions raised from the research carried the group decided to manually check the data set for more inconsistencies to ensure a complete data set was created, it was fortunate that this task was undertaken as there were images assigned to the wrong classifiers and would have been labelled as the wrong characters.

Due to the nature of the project the group decided that an incremental approach to development would be appropriate as it would allow for scaling and give time to train and change neural networks.

Milestones were set to ensure continuous development and research:

1. Develop a neural network to recognise the MNIST data set.
2. Develop a CNN to recognise the MNIST data set.
3. Acquire the merged file from the NIST data set and restructure it.
4. Assign labels to the data set to allow for training and testing.
5. Begin training the CNN to recognise the data set created from NIST.
6. Achieve an accuracy above 90 percent when testing neural network.
7. Create a web API capable of taking user input and outputting a prediction.

### 5.5.2 Version control

Having used github in previous academic years it was the obvious choice in order to keep track of source code. Due to having to create a data set from the NIST data set and then having to validate it, this meant the development of the final model and web API started later than anticipated. The division of work between both group members is as follows:

- Kimberly Burke: Development of the web API and develop the hosting service.
- Niall Devery: Development of the machine learning aspect and creating and validating the data set to be used.
- Both members researched relevant areas of interest relevant to the project, documented progress throughout the project and are involved in the write up of the dissertation of the section in which they researched.

# Chapter 6

# System Design

In the section we will be discussing the system architecture and design as well as some of the design decisions made throughout the project and giving a brief overview of hoe snippets of code work.

## 6.1 System Architecture

Throughout the course of this project there were a number of changes made from the original system architecture from the initial architecture depicted in *Figure 5.1.* The restructuring of the project was some what major and did greatly benefit the project, with on-going research yielding results design decision will be discussed under the following headings: *Web API, Machine learning, data set*

### 6.1.1 Web API

After researching web applications we came across a few different options. The one found that best suited our project needs was flask. Flask and python work hand in hand and provide a very neat web application. However the main reason we opted with Flask was its core ability to integrate a trained neural network into its system. It provided the functionality to load in a trained model when converted to a h5 file, then on the user side an image could be uploaded which was then stored into a folder and the image would then be send into the trained model. The model would receive the image and make a prediction, it would then output the results through the web API and display them to the user.

### 6.1.2 Data Set

After conducting research into the NIST data and discovering that the predefined training and test set were not representative of real world data the decision was made to create a data set comprised of samples taken from the NIST data set and assign labels to these images by using the file names. A python script was used to rename the images to achieve this.

```
import os

os.chdir('Path/Images')

i = 0
label = "A"
print(os.getcwd())

for f in os.listdir():
        os.rename(f + "_" + label + "_" + i.str())
        i++
```

The label would have be changed manually and the images would be segregated by which character they represented once the images were renamed they could all be put into the same folder creating the new data set. Tests were created at first to ensure that the images named successfully.

### 6.1.3 Machine learning

The machine learning section of this project was built using Python 3, Tensorflow, Keras and Numpy. Machine learning is the study of algorithms and statistical models that computer systems use to effectively perform a task without being specifically programmed to do so but rather "learning" through inference and pattern recognition.

```
from PIL import Image
import glob
import numpy as np
import os
from os.path import basename

image_label_list = []
for filename in glob.glob('data_set/*.png'): #assuming png
    im = Image.open(filename)
    label = (filename.split("_")[1])
```

```
    #print(label)
    image_label_list.append(label)
    im.close()
```

The above code iterates over all of the files in the directory ending with the ".png" file extension in the data set taking the first character after the first underscore encountered and appends it to a list variable which will contain all of the labels for the training and test set. This is carried out as part of preprocessing.

```
flat_Image_List = [] # new list containing flatten image arrays
for filename in glob.glob('data_set/*.png'):# iterate over the images
    im = Image.open(filename)
    img = np.reshape(im, (12288))
    flat_Image_List.append(img) # Append to the list
    #plt.imshow(img)
    im.close()
```

The above code iterates over the same directory in the same manner and opens each images and reshapes them and appends them to a list which will be used to train and test the neural network.

Neural networks typically only deal in numbers even the labels in the data set correspond to numbers this is achieved by using scikit learn [16]. Scikit learn allows developers to convert the labels in a data set to integer using a label encoder. This is done by having each class of labels corresponding to an integer. The interesting thing with scikit learn is the One Hot Encoder which converts the integers created by the label encoder to a form that can be easily interpreted by the neural network. The *Figure 6.1* show the label data under going the changes from a label encoder to a one hot encoder. The format and length the data takes once it has been converted to one hot encoding is determined by the number of classifiers in the data set in this case it is fifteen. Each one of these array of numbers correspond to a label for example [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] corresponds to a 0 in the labels. This is easy for the neural network to use because the final layer or output layer will consist of a number of neurons that is determined by the number of possible predictions it can make. The activation of the neurons can be compared to the one hot encoder equivalent of a label and this is how the neural network can make prediction based off of numbers.

The visual cortex is the part of the human brain that processes visual information. The structure of the visual cortex serves as the inspiration as to how CNNs work. The visual information is passed from one cortical area or "layer" to the next with each one detecting something more specific about the information as it progresses throughout these "layers". The same

```
label_encoder = LabelEncoder()
templabelData = label_encoder.fit_transform(data)
print(templabelData)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

```
# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
labelData = labelData.reshape(len(labelData), 1)
labelData = onehot_encoder.fit_transform(labelData)
print(labelData)
```

```
[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

```
print(labelData[68000])
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```

Figure 6.1: Encoding

```
from sklearn.model_selection import train_test_split
# create training and testing vars
X_train, X_test, y_train, y_test = train_test_split(image_array, labelData, test_size=0.15)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(62096, 64, 64, 3) (62096, 15)
(10959, 64, 64, 3) (10959, 15)
```

Figure 6.2: Caption

can be said for the process in which CNNs recognize information. Through research and some trial and error the group discovered that the most effective approach to take to large scale image recognition was to use a CNN.

```
# Import keras.
import keras as kr
import tensorflow as tf

# Importing the required Keras modules containing model and layers
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
# Start a neural network, building it by layers.
model = kr.models.Sequential()
# Initial convolutiona layer.
model.add(Conv2D(64, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected la
model.add(Dense(256, activation=tf.nn.relu))
model.add(Dropout(0.4))
model.add(Dense(50,activation=tf.nn.relu))

# Add a 15 neuron output layer.
model.add(Dense(15, activation='softmax'))
```

Seen above is the code involved in creating the CNN.

As we can see in *Figure 6.3* each epoch of the CNN took some time to train due to the size of the images being inputted but the accuracy was greatly increased because of the convolutional aspect of the model.

The data set and labels were randomized to avoid overfitting and then were split into two sets one for training and one for testing. The model would be fed the training set and the test set would be unseen until the model had completed the number of epochs (Number of times the training

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Number of Epoch is the amount of times the training set is put through the model
# The batch size is the amount of images the models processes at one time
model.fit(X_train,y_train, epochs=10, batch_size=100)
```

```
Epoch 1/10
62096/62096 [==============================] - 642s 10ms/step - loss: 1.5527 - acc: 0.5350
Epoch 2/10
62096/62096 [==============================] - 682s 11ms/step - loss: 0.5212 - acc: 0.8390
Epoch 3/10
62096/62096 [==============================] - 704s 11ms/step - loss: 0.2924 - acc: 0.9099
Epoch 4/10
62096/62096 [==============================] - 707s 11ms/step - loss: 0.2310 - acc: 0.9280
Epoch 5/10
62096/62096 [==============================] - 684s 11ms/step - loss: 0.1931 - acc: 0.9390
Epoch 6/10
62096/62096 [==============================] - 680s 11ms/step - loss: 0.1707 - acc: 0.9461
Epoch 7/10
62096/62096 [==============================] - 682s 11ms/step - loss: 0.1538 - acc: 0.9515
Epoch 8/10
62096/62096 [==============================] - 681s 11ms/step - loss: 0.1511 - acc: 0.9522
Epoch 9/10
62096/62096 [==============================] - 656s 11ms/step - loss: 0.1451 - acc: 0.9550
Epoch 10/10
62096/62096 [==============================] - 616s 10ms/step - loss: 0.1233 - acc: 0.9611
```

Figure 6.3: Model Training

set is inputted the model to train it) and then the model is then tested. The group decided to use a split of 85 - 15, meaning the test set is comprised of 15 percent of the original data set and the rest was used for the training set as seen in *Figure 6.2*.

# Chapter 7

# System Evaluation

In this section we will be evaluating the project based upon the objectives set out in *section 2.7* and discussing the methods in which the project was tested as well as conveying problems and complications encountered throughout the project life cycle.

## 7.1 Results of the Application

The results of our application were successful on a smaller scale than we had initially anticipated. We built a fully functional Convolutional Neural Network thanks to our research on a small training data set instead of the full set of data we intended to build up to as part of our incremental approach. However we got a core part of our project successfully processing the NIST data set. We found challenges with the NIST data set and there were also a lack of resources compared to the MNIST data set, but we overcame these issues and successfully trained our CNN on the NIST data set.

We have expanded our understanding of machine learning and techniques involved with image recognition and data processing through the development of the image recognition.

We successfully hosted our project through an API and integrated the trained model to this system. After converting the model to a h5 file, python flask welcomes model integration and we did just. Our API consisted of a web page where the user could upload a photo and click the prediction button. This would then send the image to the model and query the model for a prediction, the results would then display to the screen for the user to see. Due to Python flasks and Python 3s coordinated relationship we were capable of allowing a user to upload an image and have it saved in the project system.

The CNN ability to recognise a users input greatly depends on the users input, not only in the case of whether it is legible or not but also the quality of the image. As previously mention in [6] the quality of an image when passed through a neural network has adverse affects on the outcome.

## 7.2 Testing

To test the API we input a range of images differing in size colour and content. We also used some of the images that the model was trained with to ensure that it was in fact able to predict the class of image once it had been embedded into the API. The API will take any image uploaded in any size.

In addition to the testing done immediately after the CNN was trained, it was tested once it was embedded into the web API by drawing character and inputting into the web API. We done this repeatedly with images of different colours and sizes. We found that images above the 500X500 pixel range we unrecognizable to the CNN. We believe this is partially due the preprocessing the inputted images go through before they are fed into the CNN where they are resized to 64X64 pixels but also due to the Max Pooling layer. As we had conducted research into Max Pooling we discovered that it is a form of compression so this can further distort an image that has been compressed to a fraction of its original size therefore result in the classifier being unable to identify the images.

## 7.3 Complications

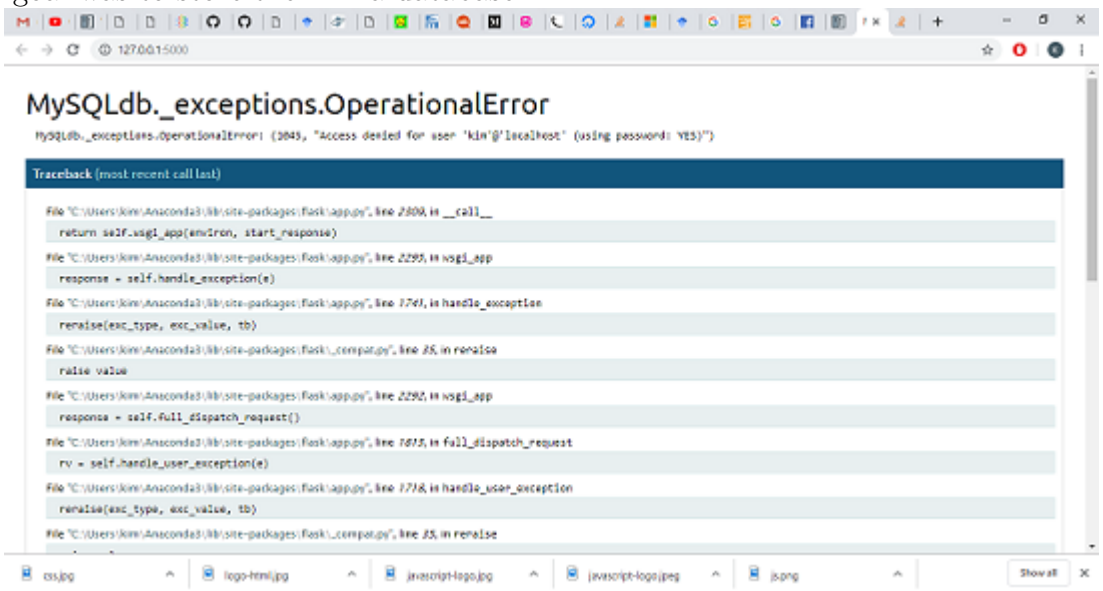
NIST data set had wrong characters in the wrong folders and as mentioned was an improper data set as was previously mentioned therefore causing further time to be consumed ensuring that the data set was of a proper quality.

After conducting research into the different methods of feeding a neural network input and experimenting with different files in which the data set could be formatted such as .csv and bitmaps after a certain amount of trial and error the format that worked was simply converting the images to numpy arrays, this approach seemed to be very computationally heavy in that the conversion would have to be completed each time the training took place. We had hoped to that once the conversion had taken place we could store the array into a file which could be read by the CNN to avoid repeating the same process over and over but due to other issues taking priority this was

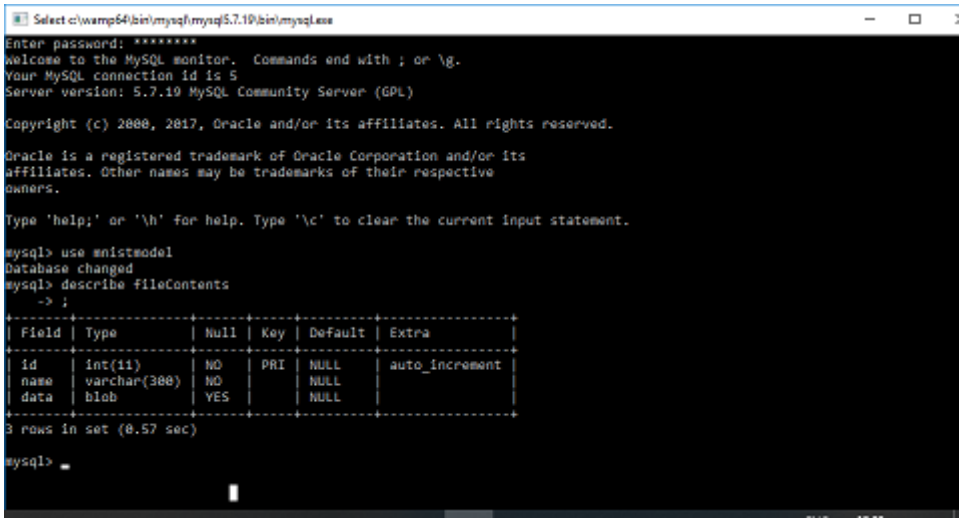
never achieved.

When developing neural networks, when switching from the MNIST data set to our own data set problems were encountered due to the the nature of the images. Even though the images only contained black and white pixels the models previously developed were made for grayscale images, as we had planned to allow users to upload their own images we had to account for possible colours of text to be uploaded. The RGB scale caused issues due to the calculations involved with calculating the amount of pixels in an image. This was later realised and rectified.

A problem that occurred while building the flask system was getting it to connect to MySql database. Our plan was to save the uploaded images to a MySql database. However we encountered many problems while trying to establish the connection. We have the images storing locally to a folder but our goal was to store them in a database.




To set up the connection four aspects were needed, a host, a username a password and the name of the database. We set up all of these aspects but were still finding that we were being refused connection. Within the database the fields included id, name and data. Type id was integer which was the primary key for the table. Type name was a varchar and was for the label of the image. Type data was of blob value which was for handling the image file itself.




Another complication encountered was hosting the application on an external server. We attempted to do this using an Apache 2.4 server. There were port issues and due to time constraints we did not successfully run our application on an external server.

# Chapter 8

# Conclusion

In this section we will be discussion our findings after the completion of this project with regards to ideas for future development and overall performance as a group.

## 8.1 Findings

From the research carried out in *section 5.2* we found that there are many different methods in which image recognition can be performed and also found that there is not set algorithm to complete a certain task within Machine Learning some methods of approach are better than others because of the nature in which the data is process and on which basis the neural net is made. We believe the fact that there is not set algorithm for certain tasks in Machine Learning is because the manner in which these neural networks learn or infer a function and the composition of the neural network itself.

## 8.2 Future Development

For future development we would like to have a larger training data set and have our model trained to the larger data set for a more elaborate and accurate system as previously mentioned and possibly have a more comprehensive CNN with perhaps more than one convolitonal layer as this would aid in differentiating between the increased number of classifiers. We would like to have the finished product hosted on an external server so that an user or machine could access our application. Another future development we would like to develop is for the images to be saving to an external database such as MySql and to have the CNN retrain itself in the background once a certain amount of images have been uploaded.

# Bibliography


[1] S. Saha, "A comprehensive guide to convolutional neural networks—the eli5 way." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[2] T. Liu, S. Fang, Y. Zhao, P. Wang, and J. Zhang, "Implementation of training convolutional neural networks," *arXiv preprint arXiv:1506.01195*, 2015.

[3] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[4] G. Van Rossum *et al.*, "Python programming language.," in *USENIX annual technical conference*, vol. 41, p. 36, 2007.

[5] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing*, pp. 4034–4038, IEEE, 2013.

[6] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6, IEEE, 2016.

[7] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60, pp. 53–60, Perth, Australia, 1995.

[8] P. Lokhande, F. Aslam, N. Hawa, J. Munir, and M. Gulamgaus, "Efficient way of web development using python and flask," 2015.

[9] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[10] T. Oliphant, "NumPy: A guide to NumPy." USA: Trelgol Publishing, 2006–.

[11] R. Shannon, "What is html," *Saatavissa: http://www. yourhtmlsource. com/starthere/whatishtml. html [viitattu 21.5. 2014]*, 2007.

[12] H.-S. Liang, K.-H. Kuo, P.-W. Lee, Y.-C. Chan, Y.-C. Lin, and M. Y. Chen, "Seess: seeing what i broke–visualizing change impact of cascading style sheets (css)," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pp. 353–356, ACM, 2013.

[13] J. Keith, *DOM scripting: web design with JavaScript and the Document Object Model*. Apress, 2006.

[14] M. Widenius and D. Axmark, *Mysql Reference Manual*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1st ed., 2002.

[15] C. McLeod, "A framework for distributed deep learning layer design in python," *arXiv preprint arXiv:1510.07303*, 2015.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

# Bibliography


[1] S. Saha, "A comprehensive guide to convolutional neural networks—the eli5 way." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[2] T. Liu, S. Fang, Y. Zhao, P. Wang, and J. Zhang, "Implementation of training convolutional neural networks," *arXiv preprint arXiv:1506.01195*, 2015.

[3] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[4] G. Van Rossum *et al.*, "Python programming language.," in *USENIX annual technical conference*, vol. 41, p. 36, 2007.

[5] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing*, pp. 4034–4038, IEEE, 2013.

[6] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6, IEEE, 2016.

[7] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60, pp. 53–60, Perth, Australia, 1995.

[8] P. Lokhande, F. Aslam, N. Hawa, J. Munir, and M. Gulamgaus, "Efficient way of web development using python and flask," 2015.

[9] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[10] T. Oliphant, "NumPy: A guide to NumPy." USA: Trelgol Publishing, 2006–.

[11] R. Shannon, "What is html," *Saatavissa: http://www. yourhtmlsource. com/starthere/whatishtml. html [viitattu 21.5. 2014]*, 2007.

[12] H.-S. Liang, K.-H. Kuo, P.-W. Lee, Y.-C. Chan, Y.-C. Lin, and M. Y. Chen, "Seess: seeing what i broke–visualizing change impact of cascading style sheets (css)," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pp. 353–356, ACM, 2013.

[13] J. Keith, *DOM scripting: web design with JavaScript and the Document Object Model*. Apress, 2006.

[14] M. Widenius and D. Axmark, *Mysql Reference Manual*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1st ed., 2002.

[15] C. McLeod, "A framework for distributed deep learning layer design in python," *arXiv preprint arXiv:1510.07303*, 2015.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.