



# Neural Networks & Biometrics as an advanced form of authentication

Aoife Sayers  
T00170881  
B.Sc. (Hons) Computing with Multimedia  
2018

## Abstract

Biometrics are used for identification and access control using an individual's physical or behavioural characteristics. Physical characteristics used for identification may comprise of physical biometrics such as fingerprints, the iris, face, hand, and DNA. Behavioural characteristics may include keyboard keystrokes, signatures, or voice data. Biometric authentication may be used as a form of authentication as it is difficult to fraudulently replicate these features. Artificial intelligence techniques such as Neural Networks are becoming more prevalent methods to enforce security measures. Neural networks are a system of neurons or nodes interconnected by connections containing weights. Neural Networks replicate the function of the human brain using a system of processing units called neurons and are interconnected by connections.

The aim of this study was to investigate if authenticating a user's biometric data with a neural network would improve current authentication in biometrics. This was accomplished using a dataset of grayscale fingerprints (NIST Special Database 4) to classify and identify people by their fingerprints.

A Convolutional Neural Network model was implemented in the Python machine learning library, Keras. The model trained on a 70% portion of the dataset and tested on the remaining 30%. The results of the training and test accuracy (20%) indicate that the accuracy is too low to prove convolutional neural networks can authenticate a user accurately. The project also concluded that a lot of samples were required per user and the training process is costly in terms of time and CPU usage. Further image processing techniques used for traditional methods of fingerprint recognition may be required to increase the accuracy of the neural network.

## Table of Contents

Abstract.....	2
Abbreviations.....	7
1. Chapter – Introduction.....	8
1.1. Overview .....	8
1.2. Problem Statement .....	9
1.3. Aims and Objectives.....	9
2. Chapter - Artificial Neural Network .....	10
2.1. Artificial Intelligence.....	10
2.2. Artificial Neural Networks .....	10
2.3. Neurons.....	11
2.4. Activation Function .....	11
Linear Activation Function .....	11
Sigmoid Activation Function .....	11
Binary/Logistic.....	11
Arctan.....	12
Bipolar .....	12
Tanh .....	12
Identity function/ Threshold/ Heavside Function .....	12
ReLu – Rectified Linear Units .....	12
2.5. Bias .....	12
2.6. Connection .....	13
2.7. Threshold/Step Function.....	13
2.8. Weights .....	13
2.9. Neural Network Layers.....	13
Input Layer .....	14
Hidden layers .....	14
Output Layer/Target .....	14
2.10. Learning Rules.....	14
Hebbian Learning Rule .....	14
Delta Learning Rule/ Widrow-Hoff Rule/ Perceptron Learning Rule .....	15
2.11. Epoch .....	15
2.12. Training Set, Testing Set & Validation Set .....	15

Training Set .....	15
Testing Set/ Validation Set.....	15
2.13. Neural Network Learning Strategies .....	16
Unsupervised Learning .....	16
Supervised Learning.....	16
2.14. Applications of Neural Networks.....	16
2.15. Neural Network Types .....	17
Single Layer Perceptron .....	17
Multi-Layer Perceptron/Deep Feedforward Neural Networks .....	17
Recurrent Neural Network.....	17
LSTM Neural Network.....	17
Hopfield Neural Network.....	17
Boltzmann Machine .....	17
2.16. Convolutional Neural Network.....	18
Layers .....	19
Steps of a Convolutional Neural Network .....	20
2.17. The dot product .....	21
2.18. Feed forward Propagation.....	21
Work through of an XOR in feed forward propagation .....	21
Delta Learning Rule .....	21
Mean Squared Error.....	21
Learning Rate.....	21
Momentum .....	21
Convergence.....	21
2.19. Backpropagation work through for XOR .....	21
2.20. Neural Network Libraries.....	22
NNabla by Sony .....	22
Tensorflow .....	22
Caffe .....	23
Keras.....	23
3. Chapter 3 – Biometrics.....	25
3.1. Introduction.....	<b>Error! Bookmark not defined.</b>
3.2. Unimodal & Multimodal systems.....	25

3.3.	Biometric Authentication .....	25
3.4.	Physical Biometrics.....	26
	Fingerprint Identification .....	26
	Fingerprint uniqueness .....	27
	Minutiae Features.....	27
	Fingerprint Image Pre-processing.....	28
	Hand geometry .....	30
	Palm Vein .....	30
	Retina .....	30
	Iris scan .....	30
	Facial recognition.....	31
	Signature .....	31
	Voice.....	31
	DNA .....	31
3.5.	Behavioural Biometrics .....	31
3.6.	Authentication.....	31
3.7.	Sessions .....	32
3.8.	Passwords.....	32
3.9.	Prevent Brute-Force Attacks .....	32
4.	Chapter 5 - Methodology & Design .....	33
4.1.	Research Undertaken.....	33
4.2.	Research Question .....	33
4.3.	Vision Document .....	33
	Scope/Outline .....	33
4.4.	Product Overview/Features .....	34
4.5.	Datasets.....	35
4.6.	System Architecture Diagram .....	36
4.7.	Solutions.....	37
	Project Specification .....	37
	Use Cases and Narratives.....	40
4.8.	Risk Analysis .....	42
4.9.	Project Plan .....	44
4.10.	Design Phase.....	45

Sequence Diagram .....	45
Work Completed .....	45
Screenshots .....	<b>Error! Bookmark not defined.</b>
Future Extensions.....	45
5. Chapter 5 - Implementation .....	46
5.1. Sprints.....	46
5.2. Sprint 1 - Set up environment .....	47
5.3. Sprint 2 : Dataset.....	51
5.4. Sprint 3 - Learning Tensorflow .....	54
5.5. Sprint 4 - Learning Keras .....	<b>Error! Bookmark not defined.</b>
5.6. Sprint 5 - Developing a Convolutional Neural Network in Keras	<b>Error! Bookmark not defined.</b>
6. Chapter 6 – Results and Conclusions .....	47
References .....	66
Appendices.....	70
Work through of an XOR in feed forward propagation .....	70
Backpropagation work through for XOR .....	71

## Abbreviations

ANN – Artificial Neural Networks

AI – Artificial Intelligence

ML – Machine Learning

RNN – Recurrent Neural Network

CNN – Convolutional Neural Network

CovNet – Convolutional Neural Network

NIST – National Institute of Standards & Technology

ITL - Information Technology Laboratory – research laboratory within the NIST

NBIS - NIST Biometric Image Software

OWASP - Open Web Application Security Project

ReLU – Rectified Linear Units

RGB – Red Blue Green

KDD – Knowledge Discovery in Databases

# 1. Chapter 1 – Introduction

## 5.1 Overview

In a world where the complexity requirements for passwords are everchanging and security breaches are becoming more commonplace, the need for more advanced methods of authenticating a user is clear. One method of authenticating a user more thoroughly is by using biometrics. Biometrics are used for identification and access control using an individual's physical or behavioural characteristics. Physical characteristics used for identification may comprise of physical biometrics such as fingerprints, the iris, face, hand, and DNA. Behavioural characteristics may include keyboard keystrokes, signatures, or voice data. Biometric authentication may be used as a form of authentication as it is difficult to fraudulently replicate these features.

Biometrics teamed with artificially intelligent techniques of improving current biometric authentication may be one method to overcome current failings in authentication. Neural networks are artificially intelligent processing systems consisting of neurons (processing units) and interconnected weights. Convolutional Neural Networks are efficient techniques of image classification. Convolutional Neural Networks are efficient at image classification and recognition on linear data after training on supervised data. The accuracy can then be tested on test data(unsupervised).

The research chapters focus on two main areas: biometrics and neural networks. The biometrics research chapter focuses on the types of biometrics, fingerprint biometrics and authentication. The neural network chapter focuses on the neural network architecture, concepts, and neural network types.

The Methodology chapter provides overview of what techniques and technologies will be used for the implementation, while the Design chapter specifies the requirements which includes prioritization, diagrams, and user-stories.

The Implementation chapter details all the work carried out in each sprint while screenshots of code and front-end functionality are also provided for a detailed understanding.



## 5.2 Problem Statement

The research question is as follows:

*Can Neural Networks be used to authenticate users based on biometric features?*

## 5.3 Aims and Objectives

### **Aim:**

To feed a dataset of fingerprints (biometric data) through a Convolutional Network and analyse if the accuracy is sufficient to authenticate a user

### **Objectives:**

- Perform KDD (Knowledge Discovery in Databases) on the dataset
- Pre-process the dataset
  - Read the class type of each fingerprint (W/T/A/L/R)
  - Sort each fingerprint into associated folder and rename the file
  - Split into a training and test folder (70:30 ratio)
  - Resize all the fingerprints
- Develop a Convolutional Neural Network model
- Train the model using the training data
- Test the model using the test data
- Perform further image processing and compare the results

## 2. Chapter 2 - Artificial Neural Network

### 5.1 Artificial Intelligence

Artificial Intelligence (AI) is a methodology in Computer Science using computers/machines or computer systems mirroring the actions and intelligence of humans with algorithms. AI may also be referred to as Machine Learning (ML). According to John McCarthy (founder of AI), he defines AI as “Machine Learning is a method of learning performed by a computer by using prediction algorithms (V.Chande, 2012).

### 5.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a system that is loosely based on the learning processes found in the neural networks of the human brain (Point, 2017). A neural network is a system of neurons or nodes (processing units) interconnected by connections containing weights. The neuron/ node is a processing element that takes inputs, adds weights to the connections, sums them up, adds a bias and uses the outcome as the argument for a single-valued function (transfer function which results in the neuron's output (M.Domnanovich, 2004). ANN's are like Biological Neural Networks in that a ANN acquires knowledge through learning. Neural Networks have been proved to be effective models in learning complex non-linear patterns which often exist in real-life data (towardsdatascience.com, 2017). Neural Networks can classify or recognise previously unseen and unsupervised data after the training process.

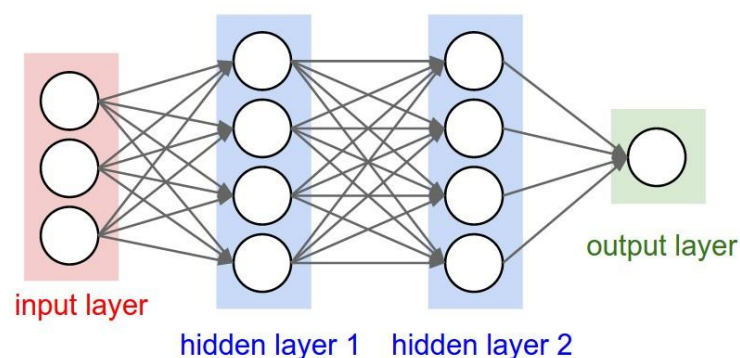


Figure 2.1 – Figure of a Multi-Layer Neural Network (cs231n, 2017)

### 5.3 Neurons

The function of an artificial neuron involves summing its weighted input signal and applying an output or activation function. (Fausset, 1994). Neurons may also be referred to nodes within a Neural Network.

### 5.4 Activation Function

Activations of a neuron is a nodes/neurons level of activity. (Fausett, 1993). The function of an Activation function is to introduce non-linearity into the network to allow a neural network to compute complex problems. (VV, 2016). The activation mathematical function inside a neuron is used to produce an output referring to its input value. The input value should exceed a specified threshold value that determines, if an output to other neurons should be generated/fired. In summary the activation function determines whether the neurons should be activated. Activation functions may also be referred to as Transfer Functions. The most commonly used Activation functions carried out by a Neuron include:

#### Linear Activation Function

Including a linear activation function has the same effect of not applying an activation function to the ANN as the input and output signals of a neuron or node remain the same. The Activation function is denoted by: (K.Vijayarekha, 2013).

$$\text{Function } f(x) = x$$

#### Sigmoid Activation Function

The Sigmoid Activation Function is a S Shaped Activation Function when graphed. The Logistic and hyperbolic tangent functions are commonly used sigmoid activation functions. Sigmoid activations functions are used for back propagation in Artificial Neural Networks (Fausset, 1994).

#### Binary/Logistic

$$\frac{1}{1 + e^{-x}}$$

Arctan

$$\text{Range from -1 to 1} \\ \frac{2}{\pi} \arctan(x);$$

Bipolar

$$\frac{2}{1 + \exp(-x)} - 1 = \frac{1 - \exp(-x)}{1 + \exp(-x)}$$

Tanh

$$\frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

Identity function/ Threshold/ Heavside Function

The Identity function is denoted as follows:

$$f(x) = x \quad \text{for all } x.$$

Single layer neural networks often use the step function to convert an input into a binary output (1 or 0) or bipolar (1 or -1). (K.Vijayarekha, 2013).

ReLu – Rectified Linear Units

The ReLu Activation Function most commonly used Activation function for deep learning models (K.Vijayarekha, 2013). It has proved to be 6 times better in convergence in comparison to the TanH layer activation function. The ReLu activation is only used within the hidden layers of the neural network. A Rectified linear unit has output 0 if the input is less than 0, When the input is positive, the derivative is just 1. The Relu Activation Function is denoted as follows: (Walia, 2017)

$$f(x) = \max(0, x)$$

## 5.5 Bias

Bias is a "pseudo" input of a neural net with any value except zero. The function of the bias is to produce different inputs for different input patterns given to the net. The bias is an

additional parameter which is used shift the output to the left or right along with the weighted sum of the inputs to the neuron.

## 5.6 Connection

A connection is a path from one neuron to another to transfer information. These are also called synapses in biological neurons and they are often parameterised with weights (Hackernoon, 2017).

## 5.7 Threshold/Step Function

The threshold value used in some activation functions determines the unit's output. Changing the threshold's purpose is to shift the graph of the activation function right or left. The threshold/step function has the same effect as a bias. (Ciaburro & Venkateswaran, 2017)

## 5.8 Weights

Weights help to determine the strength of the signal that is transferred. Weights are parameters of integer values on the connections of a neural network. Setting the weight value distinguishes in different types of neural networks. During the training process of a neural network in Backward propagation - the weights are adjusted. The adjustment of weights continues until the individual or total errors in the responses exceed a specified level or until there are no measurable errors. This is known as convergence. (Fausset, 1994)

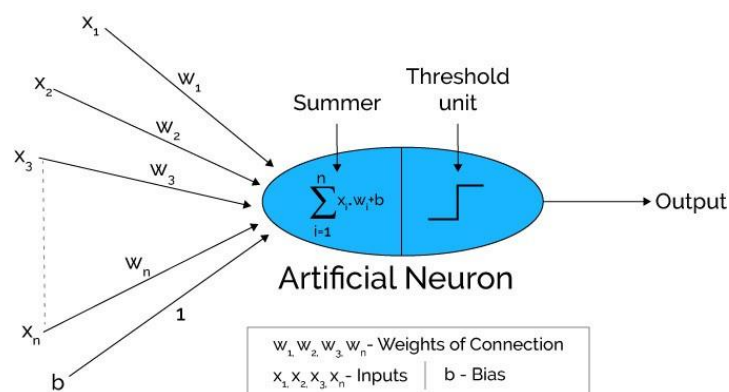


Figure 2.2 Artificial Neuron with weights of connection, inputs, and bias (Hacker Noon, 2017).

## 5.9 Neural Network Layers

A Neural Network is composed of various layers: an input layer, one or more hidden layers and Output layers – depending on the type of neural network. At each of the layers there are many interconnected neurons which contain the various Activation Functions. Each of

the connections connected to the Neurons contain weights. (To be continued below). A layer is a pattern of weighted connections between two slabs of neurons and may also function in the same way. (Fausset, 1994)

#### Input Layer

The input layer is the first layer/slab of a neural net that accepts certain input patterns and generates output values to the succeeding weight matrix. An Input layer typically transmits the input signal to all connected neurons (Fausset, 1994). In a Convolutional Neural Network, typically the input is an image.

#### Hidden layers

A Hidden layer makes sense of complex patterns, like image recognition. The function of hidden layers allows a neural network to learn logical and more complex operations. There are several different types of layers in the hidden layer of a Convolutional Neural Network. The layers include: Convolutional, Pooling and fully connected layers.

#### Output Layer/Target

An Output layer is the last slab of neural Network. An ANN outputs a value or a matrix of values (pattern), generated by the neurons of the Artificial Neural Network. The output layer is used to measure the current error value/margin of error of the net. The output layer of a Convolutional Neural Network is the class/label of image entered and the associated accuracy.

### 5.10 Learning Rules

As an Artificial Neural Network is a complex adaptive system, the weights are adjusted during training. Methods used to adjust weights are called learning rules. Learning Rules are mathematical algorithms to improve the Neural network performance. There are various learning rules for Neural Networks such as: Hebbian Learning Rule, Perceptron Learning Rule, Delta Learning Rule (Widrow-Hoff Rule) and the Competitive Learning Rule (Winner Takes All). (Point, 2017)

#### Hebbian Learning Rule

One of the first learning rules developed in 1949 was the Hebbian rule. The Hebbian learning rule was used to identify how to improve the weights of neurons of an artificial neural network. The Hebbian learning rule assumes that – If two neighbour neurons are activated and deactivated together, the weight values of the affected neurons should

increase. For neurons working in opposite phases, the weights between the affected neurons should decrease. Otherwise in the event of no signal correlation, the weight should not change. (Team, 2017) The Hebian Rule for unsupervised learning is as follows:

$$W_{ij} = x_i * x_j$$

#### Delta Learning Rule/ Widrow-Hoff Rule/ Perceptron Learning Rule

The Delta Learning rule is the most common supervised learning rule in Artificial Neural Networks. It is a gradient descent learning rule for updating the weights of the inputs to hidden layers in a neural network. “Delta” is the difference between actual and desired output. The Delta Learning also known as the Perceptron Learning Rule or the Widrow Hoff Rule.

#### 5.11 Epoch

An epoch refers to the number of iterations of providing the neural network with an input and updating the Artificial Neural Networks weights. A standard Neural Network may have in the region of thousands of epochs for sufficient training. (Ciaburro & Venkateswaran, 2017)

#### 5.12 Training Set, Testing Set & Validation Set

##### Training Set

The Training set of a neural network contains classes of data (images etc) and associated labels. For example, the classes involved in this project are the different fingerprint types: Whorl, Loop, Tented Arch, Right Arch and Arch. The purpose of a training set is to increase the performance accuracy. A sufficient amount of data should be fed through the network for Training a neural Network. A train test split should be implemented on shuffled data. A common percentage split for a train set is between 70% to 80%.

##### Testing Set/ Validation Set

Once a neural network is trained, it is often tested/validated to measure the neural network’s accuracy. The test set should ideally be unsupervised (contain no labels). The validation sets function is to reduce the likelihood of overfitting of the neural network. The validation set verifies that any increase in accuracy of the training data includes and increase

in accuracy in unseen data by the neural network. A train test split should be implemented on shuffled data. A common percentage split for a test set is between 20% to 30%.

### 5.13 Neural Network Learning Strategies

There are 2 learning strategies for Neural Networks: Supervised Learning and Unsupervised Learning

#### Unsupervised Learning

In Unsupervised learning techniques no target patterns exist. An example of unsupervised learning technique may be a dataset without known answers. Unsupervised Learning typically uses unsupervised learning technique algorithms for self-organizing neural networks. Unsupervised Learning algorithms are thought to be “more closely aligned with what some call true artificial intelligence” (Castle, 2017) An ANN determines structures & patterns in data. A cost function alerts the neural network to how much it is deviating from its target. The network then adjusts its weights for each iteration/epoch of the neural network (AppliedGo, 2016).

#### Supervised Learning

Supervised Learning techniques have an output pattern of the network which is compared with target output pattern. They are usually composed of a large set of test data with known results. Each iteration of the dataset, the target is compared with the output result and the weights are adjusted. For example, the teacher feeds student or the ANN with some example data about which the teacher already knows the answers. (Applied Go. 2017). Supervised Learning algorithms linear and logistic regression, multi-class classification, and support vector machines (Castle, 2017). The majority of Artificial Neural Networks use supervised learning techniques.

### 5.14 Applications of Neural Networks

Neural Networks have been applied across a range of industries. In financial industries neural networks have been utilised in stock market forecasts, prediction of bankruptcy, credit worthiness and detecting frauds. In medical industries applications of neural networks have been used in diagnosing data intensive diseases in detecting cancer (Filippo Amato, 2013).



Neural Networks appear to be used most frequently in data mining applications for prediction, classification, identifying change and deviation and detection. (Alyuda.com, 2015).

### 5.15 Neural Network Types

The various Neural Network types include:

#### Single Layer Perceptron

A single layer neural network is composed of 2 or more input nodes and one output node. (Hackernoon, 2017)

#### Multi-Layer Perceptron/Deep Feedforward Neural Networks

A Multilayer perceptron is composed of 2 or more input nodes, various hidden layers and 1 or more output nodes. (Hackernoon, 2017)

#### Recurrent Neural Network

A Recurrent Neural Network is often referred to as a RNN. The hidden layer neurons in this network has self-connections. A recurrent neural network has larger memory requirements. Hidden layer neurons receive activation from the lower layer as well as it previous activation value. (Fausset, 1994)

#### LSTM Neural Network

A LSTM Neural Network is a type of RNN neural network in which memory cell is incorporated inside hidden layer neurons is called LSTM network. (Hackernoon, 2017) These neural networks allow information to persist over various iterations. LSTM is an abbreviation for Long Short Memory Neural Networks.

#### Hopfield Neural Network

This is a type of network where all neurons are connected to every other neuron via weights. The network is trained with input patterns by setting a value of neurons to the desired pattern. Then its weights are computed, however the weights do not change. Once trained a pattern, the network will converge to the trained pattern (Hackernoon, 2017).

#### Boltzmann Machine

The Boltzmann Machine network can operate with or without learning. Without learning, the weights are fixed. The network solves operations by changing the activations (1 or 0) of the units based on a probability distribution (Fausset, 1994). A Boltzmann machine is like a

Hopfield network except some neurons are input, while other are hidden in nature. The weights are initialized randomly and adjusted throughout back propagation algorithm.

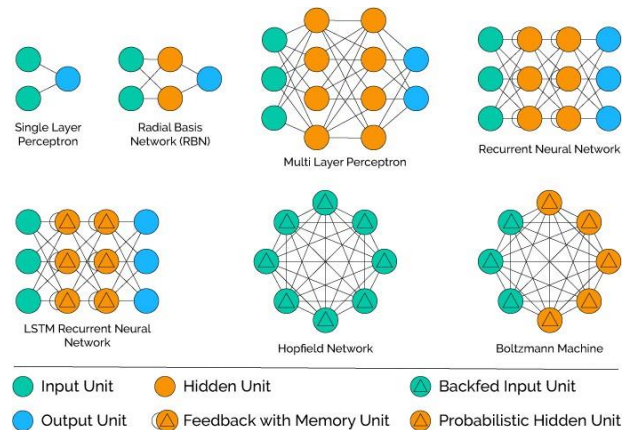


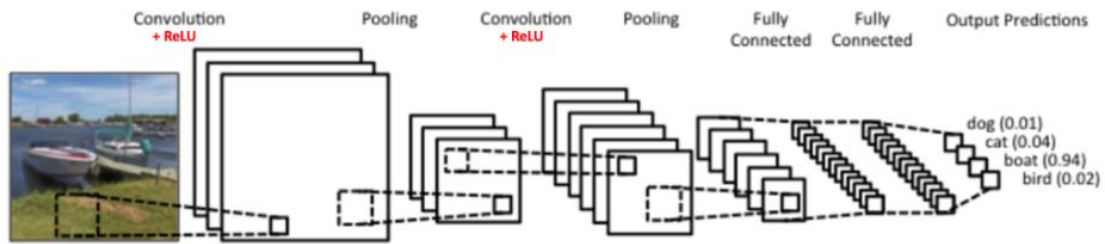
Figure 2.3 Diagrams of different types of Neural networks (Hackernoon, 2017)

## 5.16 Convolutional Neural Network

A Convolutional Neural Network may be referred to as a CNN, ConvNet or Deep Neural Network (ujjwalkarn, 2016). Convolutional Neural Networks are used for Image Recognition and classification and have a proven record in computer vision specifically, facial recognition, traffic signs detection, and object detection (ujjwalkarn, 2016). Convolutional Neural Networks are also used for voice recognition and natural language processing. A range of APIs and libraries e.g. TensorFlow and Keras are available for developing out of the box CNNs.

LeNet was the first CNN architecture developed by Yann LeCun in the 1990s (ujjwalkarn, 2016). The LeNet was developed to recognise characters as it a common task with the NIST Handwritten Digits Dataset (ujjwalkarn, 2016).

Multi-Layer neural networks) are wasteful and inefficient at image recognition as the huge number of parameters to train the network would quickly lead to overfitting. For example, to feed an image of size 200x200x3, (200w, 200h, 3 colour channels) into a standard ANN would lead to neurons that have  $200 \times 200 \times 3 = 120,000$  weights (cs231n, 2017).



*Figure 2.4 Diagram of a Convolutional Neural Network*

### Layers of a Convolutional Neural Network

A Convolutional layer is comprised of a stack of a convolutional layer followed by a pooling layer, and a dense layer (TensorFlow, 2017).

#### *Input Layer*

The input layer of a convolutional neural network takes an Image as an input. However, sometimes other types of data may be used such as audio files. An image is in an input layer is a multi dimensional array of values and can be referred to as a tensor (ujjwalkarn, 2016). The channels in a regular coloured image contain three channels for RGB (Red Blue and Green) (ujjwalkarn, 2016). For a grayscale image, 1 channel exists (ujjwalkarn, 2016). The image and number of channels must be specified during the input layer of a CNN.

#### *Convolution Layer*

The function of the convolution layer of the neural network is to perform feature extraction on an image. A 3x3 matrix referred to as a filter slides/“stride” across each pixel of the image performing the dot product (ujjwalkarn, 2016). The result of iterating through each pixel is to produce a Convolved Feature/Activation Map/Feature Map (ujjwalkarn, 2016). The CNN then learns the values produced by the filter during training. The ReLu (Rectified Linear Units) activation function is used to introduce non-linearity in the CNN (ujjwalkarn, 2016). The RELU layer performs an activation function, such as the  $\max(0, x)$   $\max(0, x)$  thresholding at zero (Jia, et al., 2014).

#### *Pooling Layer*

The pooling layer may also be referred to as subsampling, down-sampling or max pooling. The purpose of this layer is to reduce feature mapping dimensionality of the feature map (ujjwalkarn, 2016). Various types of pooling exist such as max pooling, average pooling, and sum, however, max pooling is proven to be most effective (ujjwalkarn, 2016).

### Fully Connected/ Dense Layer

The fully connected layer in a Convolutional Neural Network replicates a standard Multi-Layer Perceptron using Activation functions such as SoftMax and Sigmoid (ujjwalkarn, 2016). This layer is often referred to as a Dense layer in neural network libraries such as Keras and TensorFlow (TensorFlow, 2017). The function of this layer is to perform classification (TensorFlow, 2017).

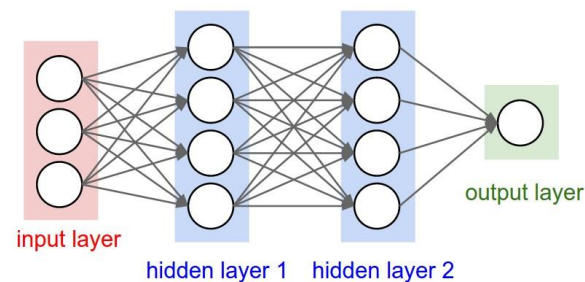


Figure 2.5 Diagram of a standard multi-layer neural network (Jia, et al., 2014)

### Output Layer

The output layer will compute the class accuracy scores (Jia, et al., 2014).

### Steps of a Convolutional Neural Network

1. Initialise the filters and weights of connections randomly
2. Training
  - a. A set of supervised training data of images with associated labels are fed into the neural network
  - b. Feed Forward Propagation occurs – image is fed from the input layer, to the convolution layer, pooling layer, fully connected layer and output layer
3. Output with accuracy of each class
4. Backpropagation is performed to calculate gradients of error.
  - a. Weights & filter values are adjusted with the purpose of increasing the accuracy at each epoch/iteration
5. The processes 1-4 continue above over the desired number of epochs on all the images in the training set
6. Testing with previously unseen images is tested on the CNN (ujjwalkarn, 2016)

### 5.17 The dot product

The Dot Product is written  $\mathbf{a} \cdot \mathbf{b}$  or the product of vectors  $\mathbf{a} \times \mathbf{b}$ . The dot product is used to calculate the Dot Product of two vectors to produce a scalar result. (Fausset, 1994). The dot product formula is shown below:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \times |\mathbf{b}|$$

### 5.18 Feed forward Propagation

One neuron layer may only have connections to neurons of other layers - e.g. Perceptron

One direction, from the input layer to the output layer.

[Work through of an XOR in feed forward propagation](#)

*Please see appendices for this work through*

#### Delta Learning Rule

The Delta Learning Rule is for minimising the square error for each training pattern

#### Mean Squared Error

The mean squared error divided by the number of output components. The mean squared error is often used for identifying the accuracy of the training and test sets.

#### Learning Rate

The learning rate parameter controls the amount by which weights are adjusted during training depending on the type of neural network. Learning rates are constant at each epoch of the neural network.

#### Momentum

Each step weight adjustments are based on current weight adjustments and the weight change from the previous iteration (Fausset, 1994)

#### Convergence

Neural Networks typically reach convergence when the weights stop changing. Convergence is a state of consistency (Fausset, 1994).

### 5.19 Backpropagation work through for XOR

*Please see the appendices for this work.*

## 5.20 Neural Network Libraries

Neural Networks are most commonly developed using Neural network libraries/APIs or frameworks. Libraries allows developers to use code in an already functional application in a stand-alone fashion. Libraries assist in abstracting the lower level more complex details of code and libraries allow for faster implementation.

### NNabla by Sony

Neural Network Libraries is a deep learning framework intended for research, development and Production on desktop PCs, HPC clusters, embedded devices and production servers. Developer can use Python and C++ APIs for developing Logistic Regression, Multi-Layer Perceptron's, Convolutional Neural Networks and Recurrent Neural Networks. However, this framework is relatively new since August 2017.

### Tensorflow

TensorFlow is a Python 3 API for machine learning and it was developed by Google's Machine Intelligence Research organisation (TensorFlow, 2017). Nodes in Tensorflow represent the various activation functions, graph edges contain multidimensional data arrays referred to as Tensors (TensorFlow, 2017). Tensors are multidimensional arrays of primitive values. The Tensor object is `tf.Tensor` (TensorFlow, 2017). TensorFlow allows multiple CPU/GPU to make computation for machine learning techniques more efficient on desktops, servers, or mobile devices (TensorFlow, 2017). TensorFlow is written with Python 3 and has many similarities to Python (TensorFlow, 2017). The TensorFlow library does numerical computation using data flow graphs. TensorFlow also has a data visualization toolkit called TensorBoard (TensorFlow, 2017). The core of TensorFlow programs include building and running a computational graph using `tf.Tensor` objects (TensorFlow, 2017). The graph describes how each tensor is computed based in each part of the graph. A `tf.Tensor` contains the following properties (TensorFlow, 2017):

- a data type (float32, int32, or string, for example)
- a shape/rank

`[2., 3., 4.]` # a rank 1 tensor; a vector with shape [3]

`[[2., 3., 4.], [5., 6., 7.]]` # a rank 2 tensor; a matrix with shape [2, 3]

`[[[1., 2., 3.], [[7., 8., 9.]]]` # a rank 3 tensor with shape [2, 1, 3]

Figure 2.17 – Examples of Tensors with Ranks - (TensorFlow, 2017)

A computational graph is a graph of nodes containing multidimensional tensors/array. Each node takes 0 to multiples tensors (Multidimensional array) as inputs and produces a tensor as an output (TensorFlow, 2017).

Estimators are high level utilities which manage TensorFlow graphs and sessions. Estimators encapsulate the following actions (TensorFlow, 2017):

- training
- evaluation
- prediction
- export for serving

Variables contain persistent memory in a TensorFlow program using a `tf.Variable` object. A `tf.Variable` object may exist outside a `session.run` call (TensorFlow, 2017).

Sessions are TensorFlow's method of running dataflow graphs across one or more local or remote instances (TensorFlow, 2017).

#### Caffe

Caffe, a deep learning framework developed with a focus expression, speed, and modularity by Berkely AI Research centre.

#### Keras

Keras is a high-level neural networks API, written in Python and can be used with TensorFlow, CNTK, or Theano. Keras was developed with a focus on enabling rapid application development. Keras is a useful deep learning library allowing for easy and fast prototype development. Keras also supports various neural network types such as Convolutional Neural Networks, Recurrent Networks etc. Keras also runs on CPU and GPU (Keras, 2017)

Keras Installation via pip

```
C:\Python27\Scripts>
C:\Python27\Scripts>pip install keras
Collecting keras
  Downloading Keras-2.0.8-py2.py3-none-any.whl (276kB)
    100% |#####| 276kB 644kB/s
Collecting scipy>=0.14 (from keras)
  Downloading scipy-1.0.0-cp27-none-win32.whl (26.4MB)
    39% |#####| 10.5MB 102kB/s eta 0:02:35
```

Figure 2.5 Installation of Keras via pip

The Sequential model is a linear stack of layers used for constructing a neural network (Keras, 2017). Layers are then added to the model. Convolutional Layers consist of 1D, 2D or 3D convolutional layers (Keras, 2017). Pooling layers include MaxPooling1D, MaxPooling2D, MaxPooling3D and AveragePooling1D (Keras, 2017). The fit(x,y) method is used to fit the images and labels to the model in the case of a CNN. The predict() accepts data from the test/validation set to test the neural network. The creation of a Keras Sequential model of a Multi-Layer Perceptron with 784 input neurons and a ReLu Activation function can be seen below:

```
model = Sequential()
model.add(Dense(32,
input_dim=784))
model.add(Activation('relu'))
```

An example of Keras's Conv2D and MaxPooling2D layers and the parameters that may be specified:

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,
dilation_rate=(1, 1), activation=None, use_bias=True, kernel_initializer='glorot_uniform',
bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None, bias_constraint=None) (Keras, 2017)
```

```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid',
data_format=None) (Keras, 2017)
```



### 3. Chapter 3 – Biometrics

#### 5.1 Biometrics

Biometrics is the measurement and analysis of physical and behavioural characteristics of an individual. Biometrics are used for identification and access control of individual's unique physical or behavioural characteristics. "The term "biometrics" is derived from the Greek words "bio" meaning life and "metric" meaning to measure". (Target, 2017)

#### 5.2 Unimodal & Multimodal systems

Unimodal systems use one physical or behavioural biometric trait. Multimodal systems use more than one physical or behavioural biometric traits for identification and verifications. (Thepade & Manade, 2017)

#### 5.3 Biometric Authentication

In recent years, biometric authentication has been growing in use. Passwords represent a dated form of authentication requiring ever increasing frustrating parameters. For example, currently most passwords require 12 characters, contain capital letters, a number, a special character etc. (BrainTree & TechCrunch, 2016).

Physical and behavioural biometrics have been proven to solve issues that traditional forms of cybersecurity do not. Malware, Social Engineering, remote access Trojan attacks have been proven to compromise the effectiveness of traditional forms of authentication.

Traditional forms of authentication include PINs, passwords, tokens, device ids, geolocation verification and so on. Biometric authentication coupled with the traditional forms of authentication may promise a more secure authentication for applications in the future.

(Turgeman, 2017) The quick succession of breaches of PINs and passwords etc are facilitating a pathway for 'Automated Biometrics'. The following factors should be considered when choosing a form of biometrics for authentication:

- Accurate Identification
- Accountability
- Easy & Safe to Use
- User Friendliness & Intrusiveness
- Security
- The physical characteristic cannot change over time

- The physical characteristic must identify a unique person

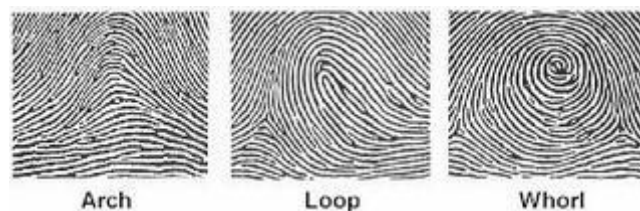
## 5.4 Physical Biometrics

Physical Biometrics systems most often use DNA, Face, Hand geometry, vein, voice, retina and iris of the eye, ear features etc.

### Fingerprint Identification

Fingerprints are formed by the fourth month of foetal development and remain the same through life, however the fingers do grow and subsequently get larger (technovelgy.com, 2015). Fingerprint biometrics are detailed, unique, difficult to alter and durable over a lifetime. A fingerprint is defined as a “pattern of ridges and valleys on a fingertip”. (Thepade & Manade, 2017). The black areas of a fingerprint are called ridges and white space in between are called valleys (Thakkar, 2015). Fingerprint patterns are most commonly divided into 3 pattern types: arches (tented arch), loops (right loop, left loop) and whorls. The loop fingerprint pattern is the most common fingerprint pattern (Shrein, 2018).

The many advantages of fingerprints as a form of biometric authentication include: very high accuracy, uniqueness, one of the most developed in biometrics, is easy to use, small database storage required, and it is standardized. However, factors which may decrease the reliability of fingerprints uses in biometrics include dry or dirty skin and age. Children’s fingerprint size may change quickly. Also, fingerprints are not identifiable from a distance, require user’s consent and may be considered intrusive by the individual.



*Figure 3.1 types of fingerprint patterns (handlines.ie, 2005)*

### Loops

Loops are fingerprint pattern that curve back on themselves to form a loop shape. Loops are the most common fingerprint pattern accounting for approximately 60 percent of fingerprint pattern types. (forensicsciencesimplified.org, 2017)

### *Whorls*

Whorls are fingerprint patterns that form circular or spiral patterns. There are four categories of whorls: plain concentric circle, central pocket loops, double loops(S shaped) and irregular shaped whorls. Whorls account for 35% of fingerprint patterns.

### *Arches*

Arches are fingerprint patterns that form wave like patterns. There are 2 categories of arches: plain arches and tented arches. Arches are the rarest type of fingerprint accounting for 5% of all pattern types. (forensicsciencesimplified.org, 2017)

### *Fingerprint uniqueness*

Fingerprints have been proven to be unique as no two people have the same fingerprints. Even identical twins, with identical DNA have different fingerprints. As fingerprints are so unique, they have been proven very effective in biometric security and forensic science. (Thepade & Manade, 2017)

### *Minutiae Features*

Minutiae refer to specific points in a fingerprint, these are the small details in a fingerprint that are most important for fingerprint recognition. The Minutiae of a fingerprint are the points where the ridges lines end or branch out in a fingerprint. (Thakkar, 2015) There are various minutiae types such as:

*Ridge ending* is where the ridge ends.

*Ridge bifurcation* is where a ridge branches out into two or more ridges.

*Ridge dots* are shorter than ridges.

*Ridge islands* are slightly longer than dots and often reside between a ridge bifurcation.

*Ponds or Lakes or Valleys* are the white spaces in between two diverging ridges. (Thakkar, 2015)

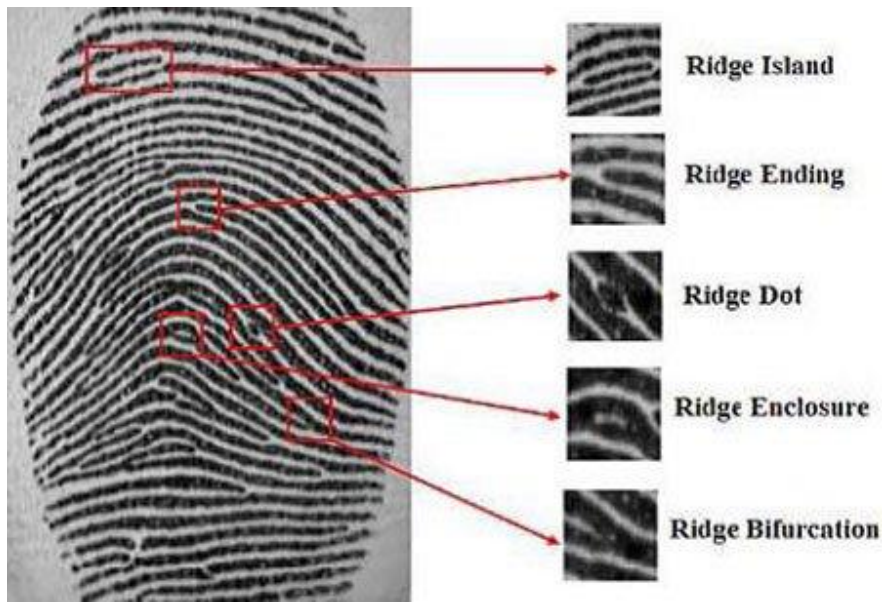


Figure 3.2 Common Minutiae patterns of a fingerprint (Thakkar, 2015)

Standard Fingerprint recognition systems consists of reading in a fingerprint via an optical sensor. The quality of the image is checked. Image enhancement and feature extraction techniques are carried out on the image. The pre-processed image is then compared with fingerprints within the database (Ali, et al., 2016).

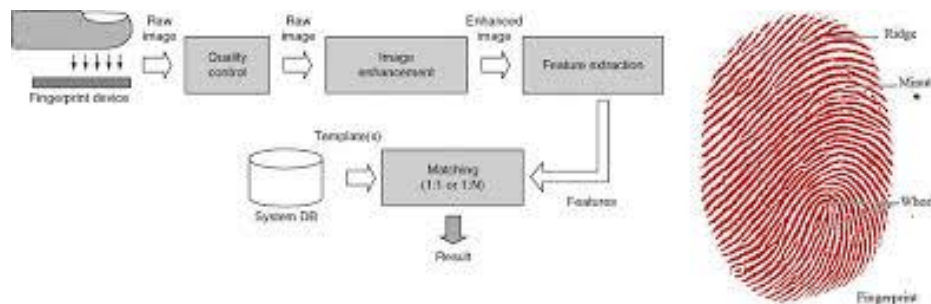


Figure 3.3 Standard Fingerprint Recognition System (ashtopustech, 2017)

Fingerprint Image Pre-processing

#### *Image Pre-processing for Fingerprint Recognition*

Image pre-processing is carried out to clean the data to a normalised format. Many steps which vary from fingerprint recognition system to system such as Image segmentation, binarization, noise removal, thinning and Gaussian filters (Ali, et al., 2016).

### *Feature Extraction for Fingerprint Recognition*

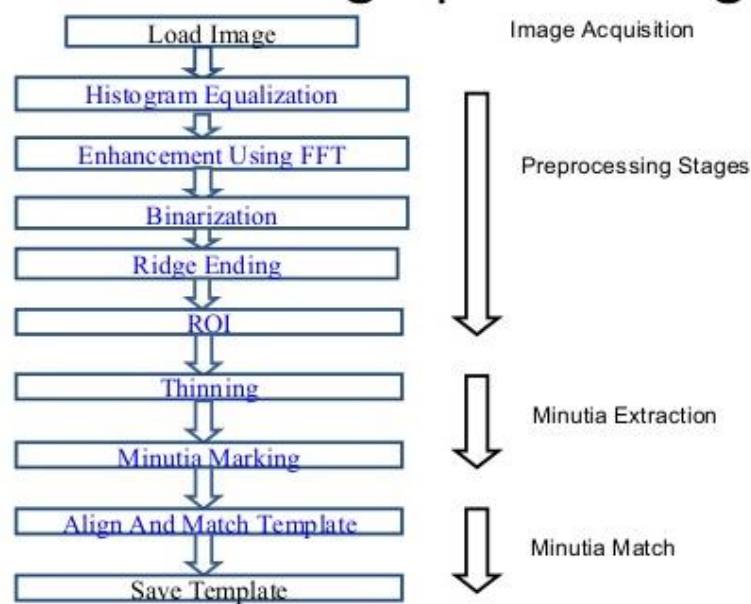
Feature extraction occurs after the image pre-processing stage. Fingerprint feature extraction typically locates measures and encodes ridge ending and bifurcations (Ali, et al., 2016). Minutiae extraction algorithm is commonly used to find the minutiae points and map their relative placement on images of fingerprints (Ali, et al., 2016). There are two types of minutiae points; Ridge ending and Ridge bifurcation (Ali, et al., 2016).

### *Feature Extraction for Fingerprint Recognition*

The fingerprint matching stage compare the pre-processed and feature extracted image of the fingerprint with the template in the database (Ali, et al., 2016).

The following image in figure 3.4 are the algorithms which take place in a fingerprint recognition system.

## Algorithms For Fingerprint Recognition



*Figure 3.4 Algorithms required for pre-processing an image of a fingerprint to perform recognition (Panda & Panda, 2014)*

### Hand geometry

Hand Geometry biometrics is based on the shape of a hand. Hand geometry considers the size of the palm, length and width of the fingers, distance between the knuckles, etc. Hand geometry data is inexpensive and not intrusive to collect. Lighting of image is not an issue, unlike obtaining fingerprint data. The hand geometry is not unique and cannot be used alone as a form of identification in identification systems. However, hand geometrics in biometrics are not unique and it is not ideal for growing children. Jewellery rings, arthritis, etc may pose a challenge in hand geometry biometrics. (360Biometrics.com, 2016)

### Palm Vein

Palm Vein biometric data is obtained using infrared beams and the veins are returned as black lines. Palm veins have been proven to have a high-level authentication accuracy due to the nature of complex patterns of the vein. Also, as vein patterns are internal, it would be difficult to forge a palm vein biometric system. (technovelgy.com, 2015)

### Retina

A retina scan analyses the capillary blood vessels located in the back of the eye using a low intensity light to take an image (technovelgy.com, 2015). The pattern of capillary blood vessels in the retina remains the same throughout a person's lifespan. The low intensity light traces a standardized path on the retina (Trader, 2012).

Retina scan data is equivalent in uniqueness to fingerprints, however, as retinæ are internal organs, they're less susceptible to modification. Also, certain eye-related medical conditions and diseases, such as cataracts and glaucoma, may compromise a retinal scan, as the blood vessels can be obscured (King, 2013).

### Iris scan

The iris is the coloured circle surrounding the pupil. The function of the Iris is to change the size of the pupil and allows different amounts of light to enter the eye. An iris scan analyses the rings, furrows, and freckles in the iris. More than 200 points of the iris scan are used for comparison. (technovelgy.com, 2015). Iris scans are acquired using infrared illumination taking images of the iris. The iris is then fed into mathematical algorithms to determine an individual's identity (Trader, 2012).

## Facial recognition

Facial recognition in biometrics constitute as the size and shape of facial characteristics and their relationship to one another. Facial recognition uses relatives between common landmarks on the face i.e. Eyes to nose to generate a unique “faceprint”. (technovelgy.com, 2015). Facial recognition is frequently used in security analysis as user consent is not required. However, facial recognition is not the most unique form of biometrics: For example, twins may have the same face. Also, human’s faces are subjective to change as they age.

## Signature

Signatures are rarely used in biometrics as a user may not sign their name the same way each time and can easily be forged.

## Voice

Voice Biometrics involve analysing the pitch, tone, cadence, and frequency of a person's voice.

## DNA

DNA is a popular biometric used in forensics and healthcare for identifying a person. Every individual has its own individual map for every cell made and can be found in everybody cell making it difficult to omit at crime scenes. DNA can be obtained via blood, hair, finger nails etc. and is unique to a person. However, obtaining DNA is an intrusive and costly process (ex-sight.com, 2009).

## 5.5 Behavioural Biometrics

Biometric that analyses users by constantly collecting information with how a user interacts with a system. Behavioural biometric examples include: keystrokes, mouse movement, hand eye coordination, pressure, handshakes, navigation, finger movements etc. (Turgeman, 2017)

## 5.6 Authentication

According to OWASP (Open Web Application Security Project), Authentication is defined as “the process of verification that an individual, entity or website is who it claims to be.

Authentication in the context of web applications is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should know.”. (OWASP, 2017)

## 5.7 Sessions

Servers are stateless meaning every request from a new client process is not maintained.

Servers cannot handle a new request if they cannot identify its origin. (Pankaj, 2017).

Session management is a process by which a server maintains the state of an entity interacting with it. The server stores the entity information to process requests and send responses. (OWASP, 2017)

## 5.8 Passwords

The application must determine the minimum length and maximum length of a password.

Passwords shorter than 10 characters are weak (NIST SP800-132). Passwords must also have a set of complexity requirements. In accordance with OWASP suggestions, a password must meet at least 3 out of the following 4 complexity requirements (OWASP, 2017):

- at least 1 uppercase character (A-Z)
- at least 1 lowercase character (a-z)
- at least 1 digit (0-9)
- at least 1 special character (punctuation) — do not forget to treat space as special characters too
- at least 10 characters
- at most 128 characters
- not more than 2 identical characters in a row (OWASP, 2017)
- Password should be changed at least every 90 days
- Passwords also should ideally not be dictionary words

## 5.9 Prevent Brute-Force Attacks

Attacker ability to guess passwords to compromise the account security.



## 4. Chapter 4 - Methodology & Design

### 5.1 Research Undertaken

Research undertaken has highlighted neural networks are a technique capable of identifying patterns. Convolutional Neural Networks may be used in biometric authentication tasks.

Neural networks contain many different components such as: neurons, activation functions, connections, bias, threshold, weights, different neural network layers, neural network learning rules, the training and testing process of a neural network, neural network learning strategies, feed forward and back propagation algorithms.

Research was also carried out into the area of biometrics. The research analysed the different types of physical and behavioural biometrics and the advantages of each.

Throughout the process of research, research concluded that physical biometrics particularly, fingerprints are the most effective in the identification of a person. Fingerprints are also unique to an individual, difficult to copy/compromise and they are the least intrusive to obtain. To train and test a neural network, a large sample of biometric data is required.

### 5.2 Research Question

*The research question is as follows:*

Using Neural Networks, can biometrics be used with neural networks as an advanced form of authentication? In addition, what type of neural network should be used to implement a biometric neural network system and do neural networks trump the traditional algorithms for detecting biometrics.

### 5.3 Vision Document

#### Scope/Outline

Biometric Neural Networks identifies patterns in a training dataset to better identify a user's biometric data e.g. fingerprints, facial recognition, and gestures etc. Biometric Neural networks will be written in Python using the Keras Neural Network library which provides abstractions and prevents the developer from having to develop a neural network from scratch. The design phase will start in September 2017 and will produce a Project Vision Statement, a Project Proposal Document, a Project Plan, Project Specification, Formal Design, Prototype, and a Design Presentation.

The following development phase will ideally produce a biometric authentication system incorporating neural networks. Implementation of sprints and the findings and conclusions will be documented also.

#### 5.4 Product Overview/Features

To prove the research question: “Neural Networks as an advanced form of authentication” involves developing a convolutional neural network to classify images. On successful completion of developing a convolutional neural network with Keras, a machine learning library. A biometric dataset of fingerprints will be fed into the neural network. A previously unseen image of a test set fingerprint will be fed through to test the accuracy. Activation functions can be used in the Neural Network to compare the accuracy.

- Locate a dataset of biometric data – fingerprints
- Identify & Learn How to an API to create an Artificial Neural
- Identify the type of Neural Network required
- Develop a Neural Network using the Neural Network API
- Integrate a dataset into the Neural Network
- Train the Neural Network
- Test the Neural Network
- Test different Activation Functions
- Improve Neural Network accuracy
- User requirements, functional and non-functional

#### **Moscow Method**

MoSCoW is an abbreviation in terms of must, should, could and would. The Moscow Method is used to develop an understanding of the requirements and priority of a project.

M – Must states that this feature is a requirement in the project to guarantee project success. S - Should states that this requirement must be included if possible, but project success does not rely on it. C – Could means that this requirement may be included if it does not affect anything else on the project W - Would like to have this requirement later, but it won't be a feature in the project delivery at this time. (Haughey, 2014)

ID	Requirement	Priority (Moscow)
001	Locate a dataset of biometric data – fingerprints	Must
002	Identify & learn how to use an API to create an Artificial Neural Network	Must
003	Identify the type of Neural Network required	Must
004	Train the Neural Network	Must
005	Test the Neural Network	Must
006	Test different Activation Functions	Should
007	Improve Neural Network accuracy	Must
008	Test Neural Network with an image within the dataset and image from online	Should
009	Use fingerprint recognition algorithms to identify whether the neural network or traditional algorithms are more accurate.	Could
010	Use Different datasets	Could
011	Research fingerprint storage security	Could
012	Read in images from biometric scanner	Could
013	Create a UI	Could

## 5.5 Datasets

A biometric dataset of fingerprints will be required for training and testing the Convolutional Neural Network:

The various datasets available are all from the NIST (The National Institute of Standards and Technology). The NIST is one of the US's oldest and most developed physical science laboratories. The NIST encompasses the ITL (Information Technology Laboratory), a research laboratory within the NIST. They supply various biometric fingerprint datasets for public use as listed below. (NIST, 2018)

The NIST Biometric Image Software (NBIS) was developed by the National Institute of Standards and Technology (NIST) for the FBI and Department of Homeland Security (DHS) in the United States. A feature of the NBIS was a neural-network based fingerprint pattern classification system called, PCASYS. PCAYS categorizes a fingerprint image into the class of arch, left or right loop, scar, tented arch, or whorl using an Artificial Neural Network. (NIST, 2018)

Special Database 4 is a dataset of 400 8-bit Grey Scale Images of Fingerprint Image Groups. The Fingerprint Groups includes 400 samples each of the five fingerprint classifications: Arch, Left & Right Loops, Tented Arch, and the Whorl. This dataset is commonly used for

fingerprint classification research with algorithms and system training and testing. (NIST, 2018)

Other Datasets provided by the NIST include:

Special Database 9 is a NIST dataset of 8-bit Grey Scale Images of Mated Fingerprint Card Pairs. (NIST, 2018)

Special Database 10 is a NIST dataset of Supplemental Fingerprint Card Data (SFCD) for NIST Special Database 9 with include rolled fingerprints and harder to find fingerprint classifications like arch, tented arch, and low count loops. (NIST, 2018)

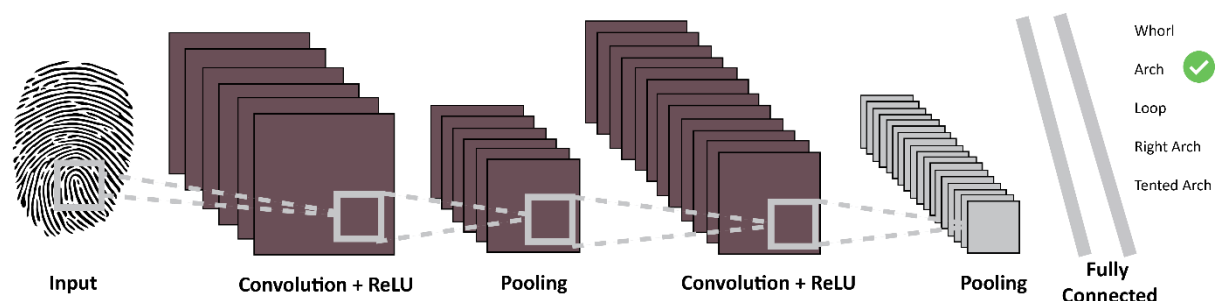
Special Database 14 is a NIST dataset of Mated Fingerprint Card Pairs 2.

This dataset includes 2,700 ten-print card pairs of rolled fingerprints (no plain impressions). (NIST, 2018)

NIST Special Database 27 and 27A is a dataset of fingerprint minutiae from latent and matching ten print images. (NIST, 2018)

Special Database 29 is a NIST dataset of Plain and Rolled Images from Paired Fingerprint Cards in 500 pixels per inch. (NIST, 2018)

## 5.6 System Architecture Diagram



*Figure 4.1 – System Architecture Diagram*

A train and test set of the NIST special database of fingerprints is fed into a Convolutional Neural Network model. The Neural network should be trained using a training set of the NIST dataset. The training will go through 2 convolutional and pooling layers of a CNN. After training the neural network, the neural network will be tested using the test set. The results should classify the fingerprint into a whorl, arch, loop, right arch or tented arch with an associated accuracy.

## 5.7 Solutions

For developing a neural network, Keras will be used. Keras is an open-source software library for Machine Intelligence. Keras is a Python language. Python 3 will be used to use the Keras library. A Convolutional Neural Network which specialise in image classification and image recognition tasks will be developed using Keras. A dataset of biometric data will be used to train the neural network. The NIST Special Database 4 is a dataset of 400 8-bit Grey Scale Images of Fingerprint Image Groups.

The CNN will be developed using PyCharm – a Jet Brains IDE.

The KDD process (Knowledge Discovery in Databases) technique will be used in the implementation of the project. The steps of the KDD process include: Data Selection, Pre-processing, Data Transformation, Data Mining and the Interpretation and Evaluation of results.

Git will be used as a version control for the code and will be hosted externally on GitHub:

<https://github.com/AoifeNicAntSaoir/BiometricCovNET>

### Project Specification

#### User Stories

ID = feature ID / user story ID

<b>ID</b>	<b>001</b>
<b>TITLE</b>	Obtain a biometric fingerprint dataset and preprocess the images
<b>DESCRIPTION</b>	As a software engineer, I want to obtain a biometric dataset of fingerprints, So that I can train and test the Convolutional Neural Network
<b>PRIORITISATION</b>	Must
<b>ACCEPTANCE CRITERIA</b>	A data set of clean data of fingerprints resized to 32x32 pixels

<b>ID</b>	<b>002</b>
-----------	------------

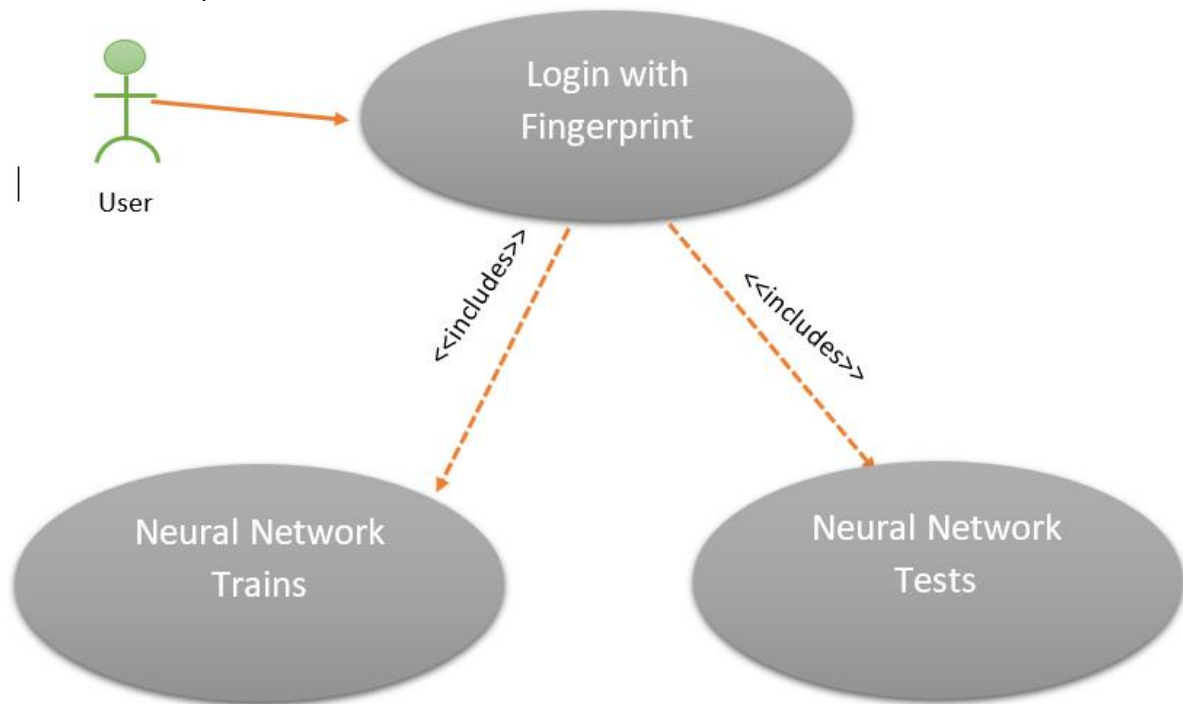
<b>TITLE</b>	Develop a Convolutional Neural Network using Keras
<b>DESCRIPTION</b>	As a software engineer, I want to develop a Convolutional Neural Network using Keras, So that I can identify fingerprints
<b>PRIORITISATION</b>	Must
<b>ACCEPTANCE CRITERIA</b>	The software engineer develops a CNN in Keras accepting an image as an input

<b>ID</b>	<b>003</b>
<b>TITLE</b>	Train an Artificial Neural Network using a training set of the fingerprint dataset by feeding it through the Convolutional Neural Network
<b>DESCRIPTION</b>	As a software engineer, I want to train a Convolutional Neural Network using Keras with a training set, So that I can build up the accuracy for the test set
<b>PRIORITISATION</b>	Must
<b>ACCEPTANCE CRITERIA</b>	The software engineer obtains a successfully trained Convolutional Neural Network.

<b>ID</b>	<b>003</b>
<b>TITLE</b>	Test an Artificial Neural Network using the test set of the fingerprint dataset by feeding it through the Convolutional Neural Network
<b>DESCRIPTION</b>	As a software engineer, I want to test a Convolutional Neural Network using Keras with a test set, So that I can accurately authenticate users by fingerprints
<b>PRIORITISATION</b>	Must

<b>ACCEPTANCE CRITERIA</b>	The software engineer obtains a successfully tested Convolutional Neural Network with a relatively high accuracy
--------------------------------	--

Use Cases and Narratives  
Use Case Description



*Figure 4.2 Use Case Diagram*



<b>Use Case Name</b>		
<b>Use Case Id</b>	0001	
<b>Priority</b>	High	
<b>Source</b>		
<b>Primary Business Actor</b>	User	
<b>Other Participating Actors</b>		
<b>Description</b>	The user will enter a biometric data – fingerprint from a data set or via a scanner. The application will be trained and use the fingerprint entered to test whether the user exists or not.	
<b>Preconditions</b>		
<b>Trigger</b>		
<b>Typical Scenario</b>	<b>Actor Action</b>	<b>System Response</b>
	<p><b>Step 1:</b> The user will launch the Biometric Neural Network application.</p> <p><b>Step 5:</b> The user will enter their biometric data.</p>	<p><b>Step 2:</b> The application will be displayed to the user upon a successful application start up.</p> <p><b>Step 3:</b> The application will train the neural network with the training data from the dataset.</p> <p><b>Step 4:</b> The application will ask the user to supply their biometric data – i.e. Fingerprint.</p> <p><b>Step 6:</b> The Neural Network will test the user entered fingerprint against the trained network and test if the user is valid.</p>
<b>Alternate Scenarios</b>	<b>Actor Action</b>	<b>System Response</b>
	<p><b>Step 5:</b> The user enters invalid biometric data</p>	<p><b>Step 6:</b> The system cannot authenticate a user</p>
<b>Conclusions</b>	The Neural Network authenticates user with biometric data.	
<b>Implementation Constraints</b>		

## 5.8 Risk Analysis

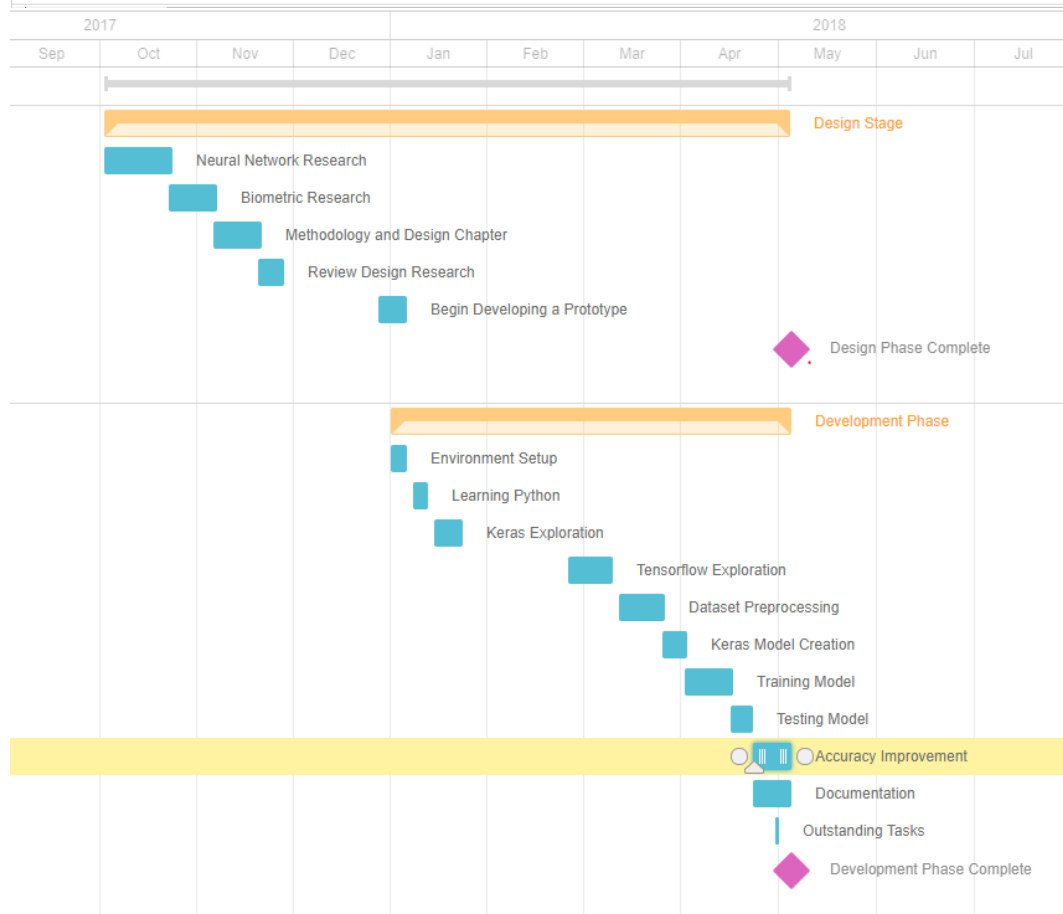
No	Risk Source	Probability			Impact			Result Prob. X Impact	Impact Areas			Risk Response
		Low 1-3	Med 4-7	High 8-10	Low 1-3	Med 4-7	High 8-10		Cost.	Sch.	Perf.	
1	Difficult learning Keras		6			7		42		x	x	Start Keras research as soon as possible. Follow Keras tutorials Read Keras documentation
2	Breakdown in communications between developer and supervisor			10			10	100		x	x	Have a clear understanding what the supervisor wants, complete work on schedule
3	Difficulty developing a Convolutional Neural Network		6			4		24		x	x	Follow tutorial and try get them working before the development phase commences
4	Difficulty finding a biometric dataset for training and testing the neural network	3					8	24		x	x	Develop own dataset of biometric data now by web scraping applications
5	Difficulty in incorporating dataset into neural network			8		5		40		x	x	Complete research and try to solve fitting of dataset into neural network
6	Neural network slow to train	3				4		12		x	x	Output training to an external file i.e. Csv file or
7	Neural Network Accuracy		4				5	20		x	x	Add more layers to improve complexity or use different activation functions or research different neural network types
8	Difficulty developing with Keras		5				7	35		x	x	Revert to a lower level neural network API with more documentation

9	Implementing a database time constraint	3				5		15		x	X	Leave the database – not a necessity when the dataset exists
10	Issues integrating a scanner		7		4			28		x	x	Drop the use of the scanner and get an image dataset from the web
11	Issues with CPU or training Neural Network		7		5			35		x	x	Use a GPU for training the neural network or train on a server
12	Dataset requires significant pre-processing to ensure clean data	2			5			10		x	x	Follow Big Data's modules steps for ensuring the dataset is properly sanitised
12	Project too simple			8			7	56		x	x	Integrate a database and secure encryption of biometric data
13	Project Over Schedule			8			7	56		x	x	Work in accordance with the project plan

## Project Plan

The work breakdown items and Gantt chart for the project plan

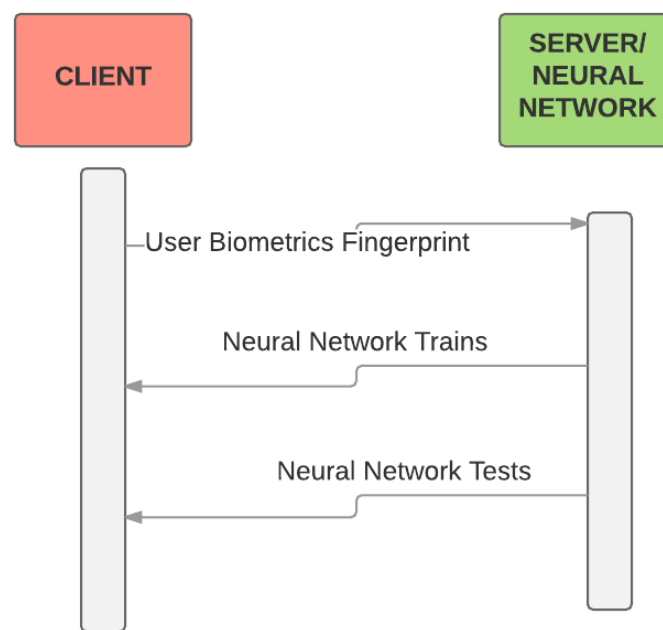
Task name	Start date	End date	Duration (week)	Progress	Estimation (hours)	Cost	Assigned	+
<input type="checkbox"/> Total estimate	02/10/2017	04/05/2018	31		797	0		
<input type="checkbox"/> Design Stage	02/10/2017	03/05/2018	30.8	0%	146	0		
Neural Network Research	02/10/2017	23/10/2017	3.2	0%	60	0	○ unassigned	
Biometric Research	23/10/2017	06/11/2017	2.2	0%	25	0	○ unassigned	
Methodology and Design Cha...	06/11/2017	20/11/2017	2.2	0%	25	0	○ unassigned	
Review Design Research	20/11/2017	27/11/2017	1.2	0%	20	0	○ unassigned	
Begin Developing a Prototype	28/12/2017	05/01/2018	1.4	0%	16	0	○ unassigned	
Design Phase Complete	03/05/2018	03/05/2018					○ unassigned	
+ Add a task + Add a milestone								
<input type="checkbox"/> Development Phase	01/01/2018	04/05/2018	18	0%	651	0		
Environment Setup	01/01/2018	05/01/2018	1	0%	40	0	○ unassigned	
Learning Python	08/01/2018	12/01/2018	1	0%	40	0	○ unassigned	
Keras Exploration	15/01/2018	24/01/2018	1.45	0%	58	0	○ unassigned	
Tensorflow Exploration	24/02/2018	09/03/2018	2	0%	80	0	○ unassigned	
Dataset Preprocessing	12/03/2018	26/03/2018	2.2	0%	88	0	○ unassigned	
Keras Model Creation	26/03/2018	02/04/2018	1.2	0%	48	0	○ unassigned	
Training Model	02/04/2018	16/04/2018	2.2	0%	88	0	○ unassigned	
Testing Model	16/04/2018	23/04/2018	1.03	0%	41	0	○ unassigned	
Accuracy Improvement	23/04/2018	04/05/2018	2	0%	80	0	○ unassigned	
Documentation	23/04/2018	04/05/2018	2	0%	80	0	○ unassigned	
Outstanding Tasks	30/04/2018	30/04/2018	0.2	0%	8	0	○ unassigned	
Development Phase Complete	04/05/2018	04/05/2018					○ unassigned	
+ Add a task + Add a milestone								
+ Add a new project								



## 5.9 Design Phase

### Sequence Diagram

The following diagram shows the sequence diagram for the Artificial Neural Network. The user enters their biometric data to the neural network. The Neural Network is trained and then tested using the client's biometric data.



*Figure 4.3 Sequence Diagram*

### Future Extensions

Further Image pre-processing stages such as binarization, ridge ending and minutiae extraction

Integrate Hardware – Use fingerprint scanners, cameras etc. instead of using a dataset for a more real-life project.

Develop a Database for the fingerprint obtained by scanners or web scraped

Ensure encryption of biometric data – encrypt biometric data i.e. Fingerprint for storage in the database

Another mode of biometric authentication – maybe facial recognition or keystrokes for characteristic biometric authentication

## 5. Chapter 5 - Implementation

### 5.1 Sprints

Task	Start Date	End Date
Environment Setup & Learn Python	1 January 2018	12 January 2018
Keras Exploration & TensorFlow exploration	15 January 2018	9 March 2018
Dataset Pre-processing	12 March 2018	26 March 2018
Keras Model Creation & Training	2 April 2018	16 April 2018
Testing Model	16 April 2018	23 April 2018
Accuracy Improvement	23 April 2018	Ongoing

## Sprint 1 - Set up environment & Learn Python

Sprint No.	Start Date	Finish Date
1	1 January 2018	12 January 2018

Task Number	Details	Status
1	Setup the Python environment of Python 3	Complete
2	Install a Python IDE – PyCharm by JetBrains Community Edition	Complete
3	Configure the Python Interpreter for a Python Project	Complete
4	Install modules for project – Keras, Tensorflow, Numpy, Pandas, OpenCV	Complete
5.	Set up local Version Control System on Git	Complete
6.	Set up remote Version Control System repository on GitHub	Complete
7.	Learn Python Basics – syntax, using modules, functions, control structures etc.	Complete

### Environment Setup

Python 3 was released in 2008 and is the successor to Python 2.7. Python is free and can be downloaded from <https://www.python.org/downloads/>.

PyCharm is a Python IDE by JetBrains. JetBrains have many other IDEs and tools such as JetBrains for Java, PhpStorm for PHP, TeamCity Continuous Integration tools etc. PyCharm offer a Professional, Educational and Community Edition of PyCharm. PyCharm Community Edition is free and open source. PyCharm's features include a Python Editor, Graphical Debugger, and a Test runner Sprint.

Python modules are scripts of Python code encapsulating functions and libraries. A module can be imported with specific functions, classes, and variables. Python modules can be imported as follows:

```
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
from keras.utils import np_utils

import shutil
import os
```

Figure 5.1 Importing modules in Python

*pip install Keras*

```
c:\Anaconda\Scripts>pip install tensorflow
Requirement already satisfied: tensorflow in c:\anaconda\lib\site-packages
Requirement already satisfied: werkzeug>=0.11.10 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: wheel>=0.26 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: markdown>=2.6.8 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: html5lib>=0.9999999 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: protobuf>=3.2.0 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: six>=1.10.0 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: backports weakref==1.0rc1 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: bleach>=1.5.0 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: numpy>=1.11.0 in c:\anaconda\lib\site-packages (from tensorflow)
Requirement already satisfied: setuptools in c:\anaconda\lib\site-packages (from protobuf>=3.2.0->tensorflow)
```

Figure 5.2 Installation of a module

```
Downloading and Extracting Packages
keras 2.1.4: ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

c:\Anaconda\Scripts>
```

Figure 5.3 Result of Installation of a module

Project: fingerprints > Project Interpreter For current project

Project Interpreter Python 3.6 Clickhouse-driver 0.0.1

Package	Version	Latest
jupyterlab_nb_ext_conf	0.1.0	
alabaster	0.7.10	0.7.10
anaconda	custom	
anaconda-client	1.6.5	1.2.2
anaconda-navigator	1.6.9	
anaconda-project	0.8.0	
aiohttp	0.22.0	0.8.2
astroid	1.5.3	0.24.0
astropy	2.0.2	1.6.1
babel	2.5.0	3.0
backports	1.9	
backports.shutil_get_terminal_size	1.0.0	
backports.weakref	1.0e1	1.0.post1
beautifulsoup4	4.6.0	4.6.0
bitarray	0.8.1	0.8.1
bkcharts	0.2	0.2
blaze	0.11.3	0.10.1
bleach	1.5.0	2.1.2
bokeh	0.12.10	0.12.14
boto	2.48.0	2.48.0
bottleneck	1.2.1	
bz2p	1.0.6	
ca-certificates	2017.08.26	
cachecontrol	0.12.3	
certifi	2018.1.18	2018.1.18
cffi	1.10.0	1.11.5
chardet	3.0.4	3.0.4
click	6.7	6.7

*Figure 5.4* Installation of modules via the PyCharm IDE



To create a project in PyCharm, click create new project and name it. Create a directory in the project called 'src'. Right click on the 'src' directory and click Mark as sources root. Create a Python file to run within the source directory.

## Learning Python Basics

As most Neural Network libraries use the language Python, it is necessary to have a good understanding of the syntax and concepts of Python. The Python documentation provided a tutorial for beginners to follow: <https://docs.python.org/3/library/index.html#library-index>. The following code snippets detail the basic concepts and syntax of Python 3.

### Importing modules

```
import time
import calendar
```

### Printing to console

```
print("Hello, Python")
```

### User input

```
raw_input("\n\nPress the key to exit\n")
print("Hello my name is " + firstName + " " + surname)
print("\nI drove " + str(miles) + " miles today")
#can't combine float and strings
age = input("\nWhat is your age?\n")
print("You said you were " + str(age) + " years old!")
```

### Variables

```
#Standard data types: Number, String, List, Tuple, Dictionary
counter = 100
miles = 1000.50
firstName = "Aoife"
surname = 'Sayers'
```

### List

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print(list)    # Prints complete list
print(list[0]) # Prints first element of the list
print(list[1:3]) # Prints elements starting from 2nd till 3rd
print(list[2:]) # Prints elements starting from 3rd element
```

### Tuple

```
print("Printing Tuples")
tuple = ("abcd", 793, 23.3, "John", 30.3)
```

### Dictionary

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
print(dict['one'])    # Prints value for 'one' key
```

### Arithmetic operators

```
a = 10
b = 20
c = 2
print(str(a) + " + " + str(b) + " = " + str((a+b)))
```

### If Elif Else

```
var = 120
```

```

if var <= 100 :
    print("The value is 100 or less")
elif var >= 101 and var <= 110:
    print("The value is greater than 100 & less than 110")
elif var >= 111 and var <= 120:
    print("The value is greater than 110 & less than 120")
else:
    print("The value is " + str(var) + " not 100")

```

## While loop

```
count = 0;
```

```

while(count <= 10):
    print("Number: " + str(count))
    count = count + 1
print("\n\t\tExited while loop")

```

## For loop

```

tables = 12
j = 0
for j in range(0,13):
    print(str(j) + " x " + str(12) + " = " + str((j*12)))

print("Number of ticks since 12 am: " + str(time.time()))
print(calendar.month(2017,3))
num = 0

```

## Functions

```

def makeTables(timesTables, HighestNum):

    for timesTables in range(0, HighestNum+1):
        print(str(num) + " x " + str(timesTables) + " = " + str(num*timesTables))
    return

makeTables(12,12)

```

## 5.2 Sprint 2 : Library Exploration: Keras vs TensorFlow

Sprint No.	Start Date	Finish Date
2	15th January 2018	9 <sup>th</sup> March 2018

Task Number	Details	Status
1	Setup the Python environment of Python 3	Complete
2	Install a Python IDE – PyCharm by JetBrains Community Edition	Complete
3	Configure the Python Interpreter for a Python Project	Complete
4	Install modules for project – Keras, Tensorflow, Numpy, Pandas, OpenCV	Complete
5.	Set up local Version Control System on Git	Complete
6.	Set up remote Version Control System repository on GitHub	Complete
7.	Learn Python Basics – syntax, using modules, functions, control structures etc.	Complete

The purpose of the sprint was to engage with two popular neural network libraries; TensorFlow and Keras. A sample project for developing Convolutional Neural Networks in both languages was implemented to become familiar with both libraries documentations and methods. As a result, the more usable and understandable documentation was used for developing the Convolutional Neural Network for this system.

This sprint focused on comparing Keras and TensorFlow's version of a CNN using the MNIST dataset. The MNIST Image CNN is the 'hello world' equivalent of every beginner's machine learning project. The MNIST dataset contains handwritten digits and associated labels. The training set contains 60,000 samples and the test set contains 10,000 examples (NIST, 2018).

### Keras NIST Convolutional Neural Network

The Keras NIST Convolutional Neural Network reached 99.25% test accuracy after 12 epochs and approximately 25 minutes. A snippet of the code used for developing the model can be seen below. Keras's code is readable and the documentation is also very detailed. Also, many tutorials for custom datasets are implemented using Keras rather than TensorFlow.

Reading in data with Keras proved to be far easier and more documented than TensorFlow's.

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
model.fit(x_train, y_train,
        batch_size=batch_size,
```

Figure 5.5 shows the sample output of the Keras MNIST CNN training. The demonstrates the training accuracy increasing at every epoch step.

```
8128/60000 [====>.....] - ETA: 13:03 - loss: 0.78200 - acc: 0.7572
8320/60000 [====>.....] - ETA: 13:29 - loss: 0.8122 - acc: 0.7403
8448/60000 [====>.....] - ETA: 13:26 - loss: 0.8056 - acc: 0.7418
8576/60000 [====>.....] - ETA: 13:24 - loss: 0.7982 - acc: 0.7441
8704/60000 [====>.....] - ETA: 13:20 - loss: 0.7918 - acc: 0.7461
8832/60000 [====>.....] - ETA: 13:18 - loss: 0.7842 - acc: 0.7484
8960/60000 [====>.....] - ETA: 13:16 - loss: 0.7776 - acc: 0.7503
9088/60000 [====>.....] - ETA: 13:14 - loss: 0.7719 - acc: 0.7526
9216/60000 [====>.....] - ETA: 13:11 - loss: 0.7656 - acc: 0.7549
9344/60000 [====>.....] - ETA: 13:09 - loss: 0.7597 - acc: 0.7571
9472/60000 [====>.....] - ETA: 13:06 - loss: 0.7524 - acc: 0.7596
9600/60000 [====>.....] - ETA: 13:04 - loss: 0.7456 - acc: 0.7622
```

*Figure 5.5 Keras NIST Training Output*

## TensorFlow NIST Convolutional Neural Network

A snippet of the TensorFlow code used for developing the CNN model can be seen below. TensorFlow's code is very verbose and technical. Reading the documentation requires a steep learning curve to understand the concepts of Neural Networks.

```
conv1 = tf.layers.conv2d(inputs=input_layer, filters=32, kernel_size=[5, 5],
padding="same", activation=tf.nn.relu)

pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

conv2 = tf.layers.conv2d(inputs=pool1, filters=64, kernel_size=[5, 5],
padding="same", activation=tf.nn.relu)

pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)

pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
```

```
INFO:tensorflow:probabilities = [[ 0.10966197  0.10436963  0.09832939  0.09742282  0.
0.10094323  0.09086428  0.09325638  0.10863883]
[ 0.10150968  0.10271324  0.08611725  0.1068951  0.10141636  0.08718498
0.10958622  0.10673322  0.08752102  0.11032295]
[ 0.10625502  0.09079039  0.08759554  0.11060774  0.09492568  0.10233439
0.10504814  0.09254416  0.09679826  0.11310072]
[ 0.0898248  0.11725562  0.09676491  0.10421059  0.09962264  0.0868704
0.10705443  0.08744232  0.10342352  0.10753073]
```

*Figure 5.6 TensorFlow NIST Training Output*

This sprint concluded that Keras would be a better option in developing a convolutional neural network due to availability of accessible documentation, the availability of external tutorial for image classification. Also, the number of research papers using Keras as a library for developing a neural network over TensorFlow was also greater.

### 5.3 Sprint 3: Dataset Pre-processing

Sprint No.	Start Date	Finish Date
3	12 <sup>th</sup> March 2018	26 <sup>th</sup> March 2018

Task Number	Details	Status
1	Source Dataset	Complete
2	Pre-processing Dataset	Complete

#### Sourcing Dataset

The dataset sourced was Special Database 4 provided by the NIST.

The NIST Biometric Image Software (NBIS) was developed by the National Institute of Standards and Technology (NIST). A feature of the NBIS was a neural-network based fingerprint pattern classification system called, PCASYS. PCAYS categorizes a fingerprint image into the class of arch, left or right loop, scar, tented arch, or whorl using an Artificial Neural Network (NIST, 2018).

The Special Database 4 dataset will be used for this project. The Special Database 4 is a dataset of 400 8-bit Grey Scale Images of Fingerprint Image Groups. The Fingerprint Groups includes 400 samples each of the five fingerprint classifications: Arch, Loops, Tented Arch, Right Arch, and the Whorl. This dataset is commonly used for fingerprint classification research with algorithms and system training and testing. (NIST, 2018). The dataset was available here <https://www.nist.gov/srd/nist-special-database-4> earlier this year but has since been removed due to a lack of documentation. The dataset will be made available within the code for this project.

#### Dataset Issue

There was an issue with the NIST fingerprint dataset for the neural network. The problem with the NIST fingerprint dataset is that there are too many classes and not enough samples. There are only 2 samples of fingerprints for each person. To authenticate someone using a CNN with only 2 samples would lead to poor accuracy. The only possible way to authenticate a user would be to authenticate a user by classifying their fingerprints by their fingerprint types. For example, a user could be one of 5 fingerprint classes: whorls, tented arches, loops, right arches, or an arch. The project will replicate the NBIS PCAYS system as detailed above.

### Pre-processing dataset Dataset

This Python script *dataSetPreprocessing.py* creates and prepares directories for the pre-processed dataset. A directory called *fingerprintsdataset/* is created and the subdirectories are created as follows:

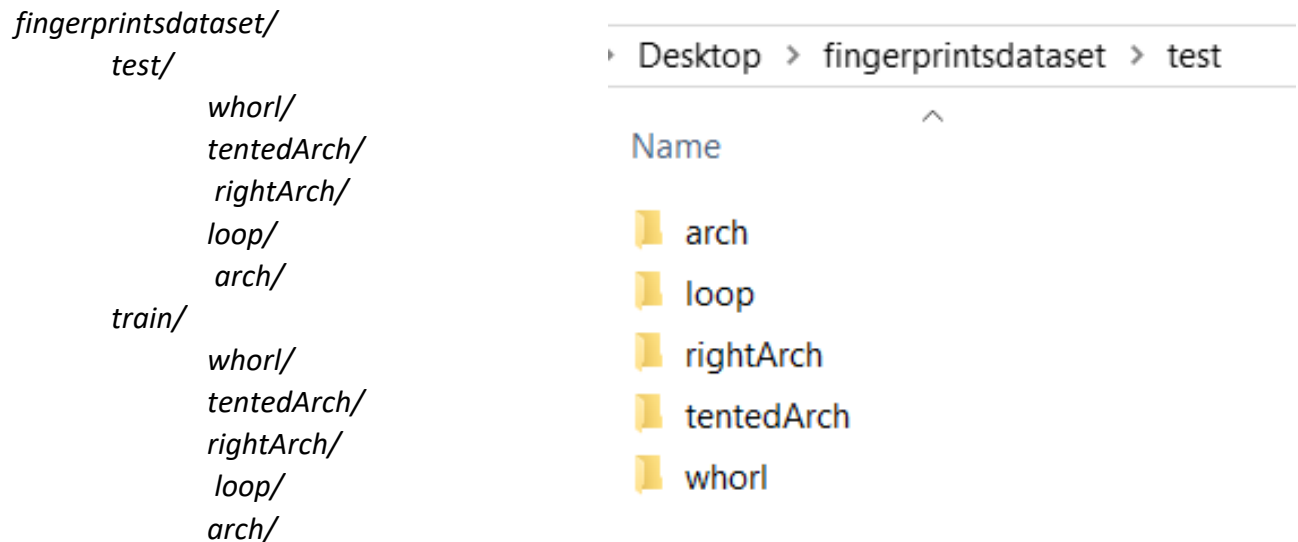


Figure 5.7 Directories created from Python script

The Python script then iterates through the folders of the NIST Special Database 4 and reads in the text file. The class/fingerprint type is then read from the text file e.g. W,T,R,L,A and subsequently added to the fingerprint type/class array.

All the arrays of the fingerprint classes/types e.g. whorls, loops etc. are shuffled and the arrays are checked to ensure there are 800 filenames in each array.

A 70% train 30% test split is created on the data. For example, the train set (70%) of the whorl should contain 560 samples and the whorl test set (30%) should contain 240 samples.

Each of the arrays of the fingerprint types take the text file name and finds the associated .png image file. The image file is copied to the directory *fingerprintsdataset/* in the correct fingerprint class directory.

After a fingerprint has been copied to it's fingerprint class directory, the image is renamed with the class name and instance of the class type for example *whorl.0.png*

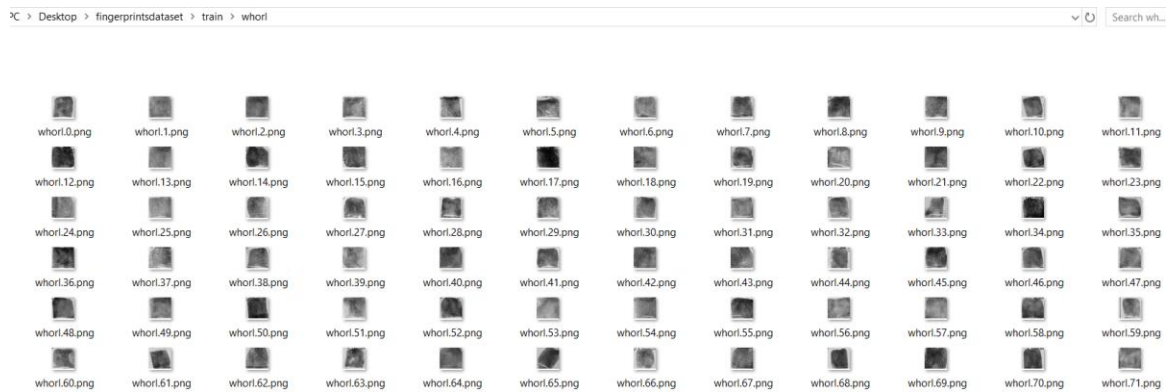


Figure 5.8 Result of dataSetPreprocessing.py on train\whorl\

```

Run dataSetPreprocessing
Classifying images...

Copying & renaming whorls
Training Set for Whorls done - now for Whorls Test Set
Done whorls

Copying and Renaming Tented Arches
Done tentedArch training set - now for tented Arch test set
Done tentedArch training set - now for tented Arch test set

Copying and Renaming Right Arches
Training set done for right arches - now for the right arches Test set
Done rightArch

Copying and Renaming Loops
finish loop train set - now for test set
Done Loop

Copying and Renaming Arches
Copying & renaming arch
Finish Arch training set - now for test set
Copying & renaming arch
Done arch

Process finished with exit code 0

```

Figure 5.9 Output of dataSetPreprocessing.py



```

whorl = []
whorlCount = 0

dir =
"../../NISTSpecialDatabase4GrayScaleImagesofFIGS/NISTSpecialDatabase4Gra
yScaleImagesofFIGS/sd04/png_txt/"
srcDir = '../fingerprintsdataset'
if os.path.exists(srcDir):
    shutil.rmtree(srcDir, ignore_errors=True)
if not os.path.exists(srcDir):
    os.makedirs(srcDir)
print('Created folder for: ' + srcDir)
if not os.path.exists('../fingerprintsdataset/train/whorl'):
    os.makedirs('../fingerprintsdataset/train/whorl')
if not os.path.exists('../fingerprintsdataset/test/whorl'):
    os.makedirs('../fingerprintsdataset/test/whorl')
print('Created folder for Whorls')
assert (len(arch)) == 800
# Train 70% Test 30%
trainSplit = int(((800 / 100) * 70))
testSplit = ((800 / 100) * 30)

print('\nCopying & renaming whorls')

for j in whorl[: trainSplit]:
    link, gone = j.split('.')
    l = link
    l += '.png'
    shutil.copy2(

'C:/Users/seamus/Desktop/NISTSpecialDatabase4GrayScaleImagesofFIGS/NISTS
pecialDatabase4GrayScaleImagesofFIGS/sd04/png_txt/' + l,
        'C:/Users/seamus/Desktop/fingerprintsdataset/train/whorl')
    m = 'whorl.' + str(whorlCount) + '.png'
    os.rename('C:/Users/seamus/Desktop/fingerprintsdataset/train/whorl/'
+ l,
            'C:/Users/seamus/Desktop/fingerprintsdataset/train/whorl/'
+ m)
print('Training Set for Whorls done - now for Whorls Test Set')
for k in whorl[trainSplit:]:
    link, gone = k.split('.')
    l = link
    l += '.png'
    shutil.copy2(
print('Done whorls')

```

*Figure 5.10 Code Snippet of dataSetPreprocessing.py*

The other Python Script created was ImageResizer.py which resizes all the images in a defined directory. The images are resized to a 32pixel width by 32 pixels in height. Anti-Aliasing is used to minimise distortion when resizing the image.

## 5.5 Sprint 4: Dataset Mining

Sprint No.	Start Date	Finish Date
4	2 <sup>nd</sup> April 2018	16 <sup>th</sup> April 2018

Task Number	Details	Status
1	Dataset to CSV File	Complete
2	Python Notebook Data Mining	Complete

### Dataset to CSV File

A script `FingerprintDS_toCSV.py` was created to read in all the genders, fingerprint classes, text file names and image file names into associated arrays. A CSV file was then created with the arrays.

```
dir =
"../../NISTSpecialDatabase4GrayScaleImagesofFIGS/NISTSpecialDatabase4GrayS
caleImagesofFIGS/sd04/png_txt/"
for filename in os.listdir(dir):
    if filename.endswith(".txt"):
        op = dir + filename
        file = open(op)
        lines = file.readlines()
        #Gender
        genderLine = lines[0]
        genderLine = genderLine.strip('\n')
        heading, gender = genderLine.split(': ')
        genderList.append(gender)
        #Fingerprint Class
        fingerprintClassLine = lines[1]
        fingerprintClassLine = fingerprintClassLine.strip('\n')
        heading, fingerprintType = fingerprintClassLine.split(': ')
        fingerprintList.append(fingerprintType)
        #TextFile and pngs
        txtFileList.append(filename)
        file, ext = filename.split('.')
        imgFileList.append(file+'.png')

df = pd.DataFrame(list(zip(genderList, fingerprintList, txtFileList,
imgFileList)),

columns=['Gender', 'FingerprintType', 'TextFiles', 'ImageFiles'])

df.to_csv('../../Fingerprint.csv', sep=',', encoding='utf-8')
```

Figure 5.11 Code Snippet of `FingerprintDS_toCSV.py`

The script above creates a CSV file: `Fingerprint.csv`.

	Gender	Fingerprint	TextFiles	ImageFiles
0	M	W	f0001_01.	f0001_01.png
1	M	R	f0002_05.	f0002_05.png
2	F	L	f0003_10.	f0003_10.png
3	M	R	f0004_05.	f0004_05.png
4	M	A	f0005_03.	f0005_03.png
5	M	A	f0006_09.	f0006_09.png
6	M	L	f0007_09.	f0007_09.png
7	F	T	f0008_10.	f0008_10.png
8	M	L	f0009_08.	f0009_08.png
9	M	W	f0010_01.	f0010_01.png
10	M	W	f0011_02.	f0011_02.png
11	F	A	f0012_01.	f0012_01.png
12	M	A	f0013_03.	f0013_03.png
13	M	W	f0014_06.	f0014_06.png

Figure 5.12 Fingerprints.csv file resulting from a python script

#### Python Notebook Data Mining

Descriptive Analytics analyse the genders of the dataset. The analysis shows there is an imbalance of genders in the dataset. There are 3250 fingerprints from Males and 750 fingerprints from females. The resulting pie chart and code of the gender analysis is demonstrated in Figure 5.13.

```

1 females = len(df[df["Gender"]=="F"])
2 males = len(df[df["Gender"]=="M"])
3 labels = ["Female", "Male"]
4 sizes= [females,males]
5 colors = ['pink','blue']
6 plt.suptitle('Percentage of Females vs Males')
7 plt.pie(sizes, labels=labels, colors=colors,
8         autopct='%1.1f%%', shadow=True, startangle=140)
9 plt.axis('equal')
10 plt.show()

```

Percentage of Females vs Males

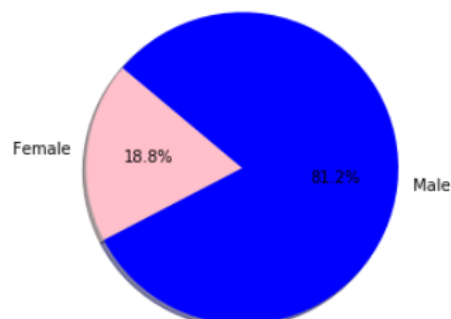


Figure 5.13 Analysis of gender in the dataset.

Further analysis demonstrates there is an equal number of each class or fingerprint type in the database. Each class e.g. whorl, tented arch, and loop all contain 800 fingerprints.

```

1 #W T R L A
2 whorls = len(df[df["FingerprintType"]=="W"])
3 tentedArches = len(df[df["FingerprintType"]=="W"])
4 rightArches = len(df[df["FingerprintType"]=="R"])
5 loops = len(df[df["FingerprintType"]=="L"])
6 arches = len(df[df["FingerprintType"]=="A"])
7
8 print('No. Whorls: ' + str(whorls) +
9       '\nNo. Tented Arches: ' + str(tentedArches) +
10      '\nNo. Right Arches: ' + str(rightArches) +
11      '\nNo. Loops: ' + str(loops) +
12      '\nNo. Arches: ' + str(arches))

```

```

No. Whorls: 800
No. Tented Arches: 800
No. Right Arches: 800
No. Loops: 800
No. Arches: 800

```

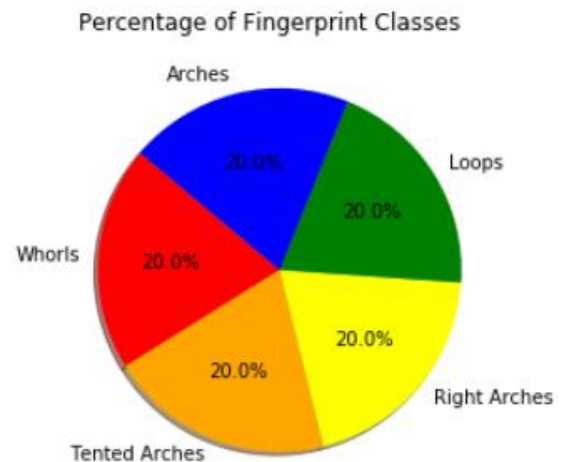


Figure 5.14 Analysis of fingerprint types in the dataset.

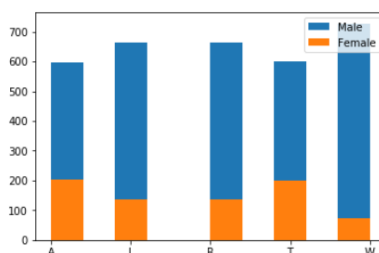
An analysis of the most common fingerprint types for each gender was also performed. The most common fingerprint type/class for females were 'A' (Arches). The most common fingerprint type for a male was the whorl. From the visualisation below, it is evident the whorl is the most common fingerprint type.

Histogram of Fingerprint types for each gender

```

1 plt.hist(df[df["Gender"]=="M"]["FingerprintType"].reset_index(drop=True), label="Male")
2 plt.hist(df[df["Gender"]=="F"]["FingerprintType"].reset_index(drop=True), label="Female")
3 plt.legend()
4 plt.show()

```



Most common fingerprint type for female

```

1 df[df["Gender"]=="F"]["FingerprintType"].mode()
0 A
dtype: object

```

Most common fingerprint type for a male

```

1 df[df["Gender"]=="M"]["FingerprintType"].mode()
0 W
dtype: object

```

Figure 5.15 – Most common fingerprint types for each gender

## 5.6 Sprint 5: Keras CNN Model

Sprint No.	Start Date	Finish Date
5	16 <sup>th</sup> April 2018	23 <sup>rd</sup> April 2018

Task Number	Details	Status
1	Develop CNN model using Keras	Complete
2	Train CNN using Training Data	Complete

### Develop CNN Model using Keras

The aim of the sprint is to replicate the image below in figure 5.16 using the Keras API. The Sequential model allows the developer pass in a stack of layers (Keras, 2017). The Convolutional layer in Keras can be implemented using Conv1D, Conv2D and Conv3D. The Pooling layers in the Keras API can be implemented using MaxPooling1D, MaxPooling2D, MaxPooling3D or AveragePooling1D, AveragePooling2D and AveragePooling3D. The Fully Connected layers in the Keras are represented using Dense Layers. The layers used for this project were two Conv2D layers, two MaxPooling2D layers and 2 Dense layers (Fully Connected layers). A tutorial in Classifying images of cats and dogs were used to construct this code: <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html> and <https://medium.com/@parthvadhadiya424/hello-world-program-in-keras-with-cnn-dog-vs-cat-classification-efc6f0da3cc5>

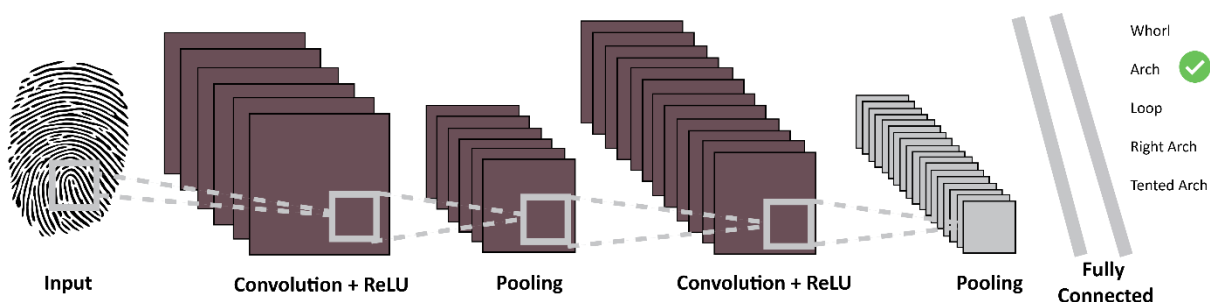


Figure 5.16 Layers required within a Convolutional Neural Network

The code below represents the Convolutional diagram in Figure 5.16. The parameters allowed for the Conv2D is as follows: (Keras, 2017)

```

from keras.models import Sequential
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Conv2D, MaxPooling2D

# Initialising the CNN# Initi
classifier = Sequential()
# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), padding='same', input_shape=(64, 64, 3),
activation='relu'))
# Step 2 - Pooling# Step
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), padding='same', input_shape=(64, 64, 3),
activation='relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Step 3 - Flattening
classifier.add(Flatten())
# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics
= ['accuracy'])
# Part 2 - Fitting the CNN to the images
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
training_set =
train_datagen.flow_from_directory('../..//fingerprintsdataset/train/',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')

test_set =
test_datagen.flow_from_directory('../..//fingerprintsdataset/test/',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')

print(test_set.image_shape)
classifier.fit_generator(
training_set,
steps_per_epoch=800,
epochs=1,
validation_data=test_set,
validation_steps=800)

score = classifier.evaluate(test_set, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

classifier.save_weights('model_wieghts.h5')
classifier.save('model_keras.h5')

```

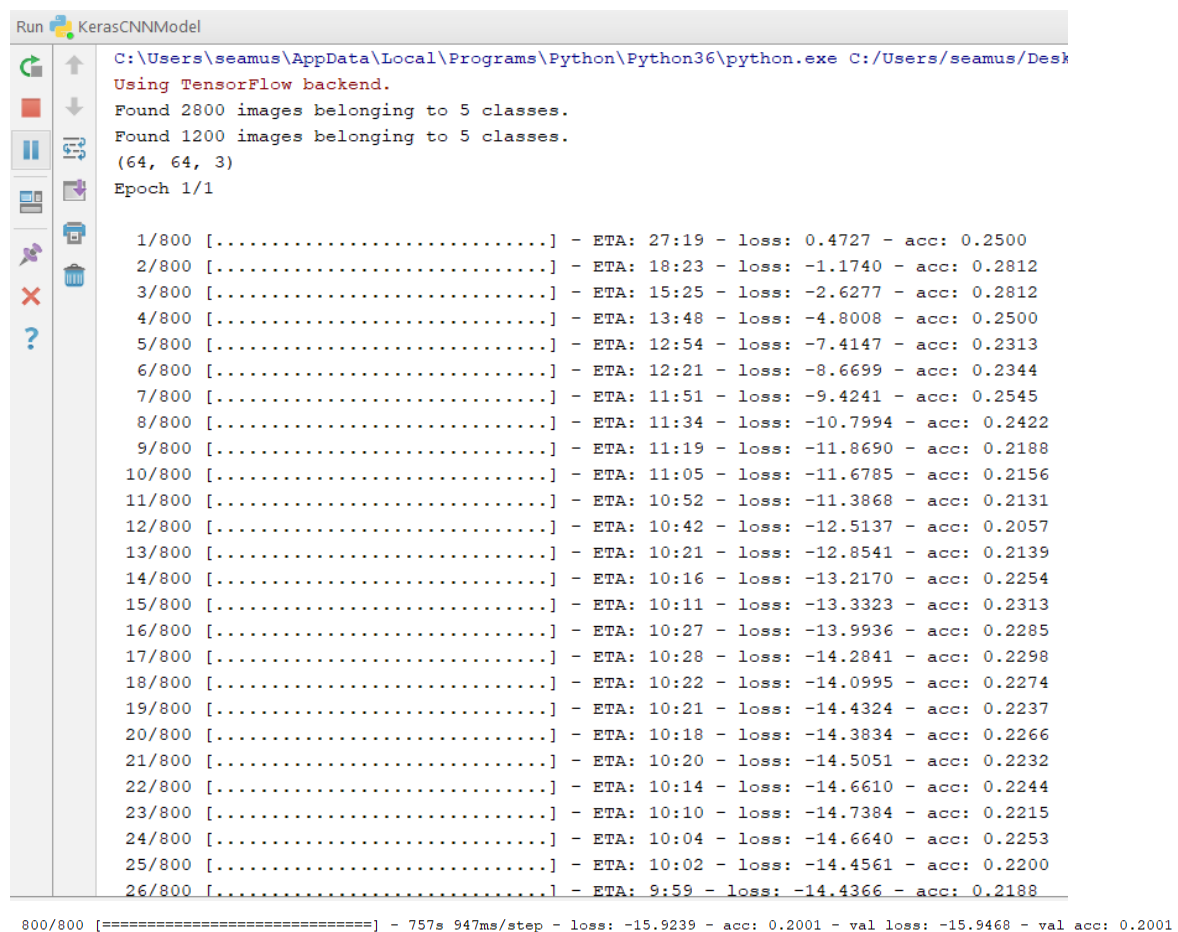
## Training the Model

To run training on the developed model, the `classifier.fit_generator` accepts the test set as a parameter, the number of epochs (iterations), steps per epoch, test set and validation steps.

The figure in 5.17 shows the running of the model and training. The Keras model reads in the training and test sets and 5 different classes.

1 epoch is only specified in this version of the model. 20 epochs are used for the most accurate training result. Each step of epoch shows a gradual increase in accuracy from 0.25 to 20%. The accuracy is not an ideal result and should be higher for authentication. The conclusion from the low accuracy is that more pre-processing for the fingerprint images must be implemented. After the training process executed, the training results and weights were saved to a h5 file.

Various models with varying numbers and layers were tested to see if it would increase the accuracy over 20%. The outcome of this was less successful.



```
Run KerasCNNModel
C:\Users\seamus\AppData\Local\Programs\Python\Python36\python.exe C:/Users/seamus/Desktop/...
Using TensorFlow backend.
Found 2800 images belonging to 5 classes.
Found 1200 images belonging to 5 classes.
(64, 64, 3)
Epoch 1/1

1/800 [.....] - ETA: 27:19 - loss: 0.4727 - acc: 0.2500
2/800 [.....] - ETA: 18:23 - loss: -1.1740 - acc: 0.2812
3/800 [.....] - ETA: 15:25 - loss: -2.6277 - acc: 0.2812
4/800 [.....] - ETA: 13:48 - loss: -4.8008 - acc: 0.2500
5/800 [.....] - ETA: 12:54 - loss: -7.4147 - acc: 0.2313
6/800 [.....] - ETA: 12:21 - loss: -8.6699 - acc: 0.2344
7/800 [.....] - ETA: 11:51 - loss: -9.4241 - acc: 0.2545
8/800 [.....] - ETA: 11:34 - loss: -10.7994 - acc: 0.2422
9/800 [.....] - ETA: 11:19 - loss: -11.8690 - acc: 0.2188
10/800 [.....] - ETA: 11:05 - loss: -11.6785 - acc: 0.2156
11/800 [.....] - ETA: 10:52 - loss: -11.3868 - acc: 0.2131
12/800 [.....] - ETA: 10:42 - loss: -12.5137 - acc: 0.2057
13/800 [.....] - ETA: 10:21 - loss: -12.8541 - acc: 0.2139
14/800 [.....] - ETA: 10:16 - loss: -13.2170 - acc: 0.2254
15/800 [.....] - ETA: 10:11 - loss: -13.3323 - acc: 0.2313
16/800 [.....] - ETA: 10:27 - loss: -13.9936 - acc: 0.2285
17/800 [.....] - ETA: 10:28 - loss: -14.2841 - acc: 0.2298
18/800 [.....] - ETA: 10:22 - loss: -14.0995 - acc: 0.2274
19/800 [.....] - ETA: 10:21 - loss: -14.4324 - acc: 0.2237
20/800 [.....] - ETA: 10:18 - loss: -14.3834 - acc: 0.2266
21/800 [.....] - ETA: 10:20 - loss: -14.5051 - acc: 0.2232
22/800 [.....] - ETA: 10:14 - loss: -14.6610 - acc: 0.2244
23/800 [.....] - ETA: 10:10 - loss: -14.7384 - acc: 0.2215
24/800 [.....] - ETA: 10:04 - loss: -14.6640 - acc: 0.2253
25/800 [.....] - ETA: 10:02 - loss: -14.4561 - acc: 0.2200
26/800 [.....] - ETA: 9:59 - loss: -14.4366 - acc: 0.2188

800/800 [=====] - 757s 947ms/step - loss: -15.9239 - acc: 0.2001 - val_loss: -15.9468 - val_acc: 0.2001
```

Figure 5.17 CNN Model compiling and testing

## 5.7 Sprint 6: Testing the CNN

Sprint No.	Start Date	Finish Date
6	19 <sup>th</sup> March 2018	30 <sup>th</sup> March 2018

Task Number	Details	Status
1	Test Keras	Complete

As the accuracy is so poor, testing is pointless. The code was used as follows to test the accuracy. The main objective of this sprint is to find methods of increasing the accuracy.

```
from keras.models import load_model
from keras.preprocessing.image import img_to_array, load_img
import numpy as np

test_model = load_model('model_keras.h5')
img = load_img('arch.560.png', False, target_size=(64, 64))
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
preds = test_model.predict_classes(x)
prob = test_model.predict_proba(x)
p = test_model.predict(x)
print(preds, prob)
print(p)
```

## 5.8 Sprint 7: Improving Accuracy

Sprint No.	Start Date	Finish Date
7	23rd	Ongoing

Task Number	Details	Status
1	Develop CNN model using Keras	Not Complete



## 7. Chapter 6 – Results and Conclusions

The results of the training set accuracy only reached 20% concluding that the accuracy is too low to authenticate a user. It is possible that more image processing algorithms would improve the accuracy. The following algorithms have been identified as methods to improving the accuracy:

- Histogram Equalization
- Enhancement using FFT
- Binarization
- Ridge Ending
- ROI
- Thinning
- Minutia Marking

The image pre-processing is a quite substantial and key phase of successful Convolutional Neural Networks.

## References

- 360Biometrics.com, 2016. *Hand Geometry Biometrics*. [Online]  
Available at: <http://www.360biometrics.com/faq/Hand-Geometry-Biometrics.php>  
[Accessed 15 Novmeber 2017].
- Ali, M. M., Mahale, V. H., Yannawar, P. & Gaikwad, A. T., 2016. Overview of fingerprint recognition system. *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference*, p. 5.
- Alyuda.com, 2015. *Neural Network Applications*. [Online]  
Available at: <http://www.alyuda.com/products/forecaster/neural-network-applications.htm>  
[Accessed 12 September 2017].
- AppliedGo, 2016. *Perceptrons - the most basic form of a neural network*. [Online]  
Available at: <https://appliedgo.net/perceptron/>  
[Accessed 12 10 2017].
- ashtopustech, 2017. *Fingerprint recognition*. [Online]  
Available at: <http://ashtopustech.com/solutions/fingerprint-recognition/>  
[Accessed 3 May 2018].
- biometricsolution.com, 2018. *Fingerprint Recognition*. [Online]  
Available at: <http://www.biometricsolution.com/>  
[Accessed 24 January 2018].
- BrainTree & TechCrunch, 2016. *Forget passwords how biometrics are transforming the security of mobile payments*. [Online]  
Available at: <https://techcrunch.com/sponsored/forget-passwords-how-biometrics-are-transforming-the-security-of-mobile-payments/>  
[Accessed 15 Novembber 2017].
- Castle, N., 2017. *Supervised and Unsupervised Machine learning algorithms*. [Online]  
Available at: <https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms>  
[Accessed 14 February 2018].
- Ciaburro, G. & Venkateswaran, B., 2017. *Neural Networks with R*. 1 ed. s.l.:Packt.
- cs231n, 2017. *Convolutional Networks*. [Online]  
Available at: <http://cs231n.github.io/convolutional-networks/>  
[Accessed 18 October 2017].
- cs231n, 2017. *convolutional-networks*. [Online]  
Available at: <http://cs231n.github.io/convolutional-networks/>  
[Accessed 30 September 2017].
- ex-sight.com, 2009. *What is "Biometrics", and what kind of convenience and security can it offer us?*. [Online]  
Available at: <http://www.ex-sight.com/biometric.htm>  
[Accessed 22 February 2018].
- Fausett, L., 1993. *Fundamentals of Neural Networks*. 2nd ed. New Jersey: Pretence Hall.
- Fausset, L., 1994. *Fundamentals of Neural Networks*. 1st ed. New Jersey: Pretence Hall.
- Filippo Amato, A. L. E. M. P.-M. P. V. A. H. J. H., 2013. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, Volume 11.

forensicsciencesimplified.org, 2017. *Principles of Fingerprint Analysis*. [Online]  
Available at: <http://www.forensicsciencesimplified.org/prints/principles.html>  
[Accessed 15 November 2017].

Fun, M. I., 2014. *Vectors Dot Product*. [Online]  
Available at: <https://www.mathsisfun.com/algebra/vectors-dot-product.html>  
[Accessed 18 October 2017].

Hackernoon, 2017. *Overview of artificial neural networks and its applications*. [Online]  
Available at: <https://hackernoon.com/overview-of-artificial-neural-networks-and-its-applications-2525c1adff7>  
[Accessed 18 October 2017].

handlines.ie, 2005. *Do you have unusual fingerprints*. [Online]  
Available at: <http://handlines.blogspot.ie/2005/09/do-you-have-unusual-fingerprints.html>  
[Accessed 15 November 2017].

Haughey, D., 2014. *Moscow Method*. [Online]  
Available at: <https://www.projectsmart.co.uk/moscow-method.php>  
[Accessed 15 February 2018].

Humphrys, D. M., n.d. *Single-layer Neural Networks (Perceptrons)*. [Online]  
Available at: <http://computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html>  
[Accessed 18 October 2017].

Jia, Y., Shelhamer, E., Donahue, J. & Sergey Karayev, J. L. R. G. S. G. T. D., 2014. Caffe: Convolutional Architecture. *Caffe: Convolutional Architecture*, p. 4.

K.Vijayarekha, D., 2013. Activation Functions. *NPTEL – Electronics & Communication Engineering – Pattern Recognition*, Volume I, p. 3.

Keras, 2017. *Keras - Github*. [Online]  
Available at: <https://github.com/fchollet/keras>  
[Accessed 18 October 2017].

King, R., 2013. *Retinal Scan Technology*. [Online]  
Available at: <http://www.biometricupdate.com/201307/explainer-retinal-scan-technology>  
[Accessed 11 November 2017].

M.Domnanovich, L. R. P. D. P. S., 2004. Prediction of trace compounds in biogas from anaerobic digestion using the MATLAB Neural Network Toolbox. *Environmental Modelling & Software*, pp. 803-810.

NIST, 2018. *NIST*. [Online]  
Available at: <https://www.nist.gov/>  
[Accessed 15 February 2018].

OWASP, 2017. *Authentication Cheat Sheet*. [Online]  
Available at: [https://www.owasp.org/index.php/Authentication\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Authentication_Cheat_Sheet)  
[Accessed 22 November 2017].

Panda, S. K. & Panda, S. S., 2014. *Fingerprint Recognition Project Slideshare LinkedIn*. [Online]  
Available at: <https://www.slideshare.net/sandeepkumparpanda/fingerprint-recognition-technique>  
[Accessed 3 May 2018].

Pankaj, 2017. *Session Management in Java – HttpServlet, Cookies, URL Rewriting*. [Online]  
Available at: <https://www.journaldev.com/1907/java-session-management-servlet-http-session-url-rewriting>  
[Accessed 22 November 2017].

Point, T., 2017. *Artificial Intelligence - Neural Networks*. [Online]  
Available at:  
[https://www.tutorialspoint.com/artificial\\_intelligence/artificial\\_intelligence\\_neural\\_networks.htm](https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm)  
[Accessed 11 September 2017].

S. Agatonovic-Kustrin, R. B., 1999. Basic concepts of artificial neural network (ANN) modeling. *Journal of Pharmaceutical and Biomedical Analysis*, Volume II, p. 11.

Shrein, J. M., 2018. Fingerprint classification using convolutional neural networks and ridge orientation images. *Computational Intelligence (SSCI), 2017 IEEE Symposium Series*, I(1), p. 8.

Stack, J. X., 2017. *Log analytics with deep learning and machine learning*. [Online]  
Available at: <https://www.xenonstack.com/blog/log-analytics-with-deep-learning-and-machine-learning>  
[Accessed 18 October 2017].

Target, T., 2017. *Biometrics*. [Online]  
Available at: <http://searchsecurity.techtarget.com/definition/biometrics>  
[Accessed 15 November 2017].

Team, D., 2017. *Learning Rules In Neural Network*. [Online]  
Available at: <https://data-flair.training/blogs/learning-rules-in-neural-network/>  
[Accessed 11 December 2017].

technovelgy.com, 2015. *Biometric Security*. [Online]  
Available at: <http://www.technovelgy.com/ct/Technology-Article.asp?ArtNum=11>  
[Accessed 15 November 2017].

TensorFlow, 2017. *About Tensorflow*. [Online]  
Available at: <https://www.tensorflow.org/>  
[Accessed 18 October 2017].

Thakkar, D., 2015. Minutiae Based Extraction in Fingerprint Recognition. *Bayometric*, 27 May, p. 1.

Thepade, S. & Manade, M., 2017. Multifinger biometric identification with score level fusion using Thepade's sorted n-ary block truncation coding. *Electronics & Telecommunication Engineering (ICWET 2016), International Conference & Workshop on*, I(1), p. 6.

towardsdatascience.com, 2017. *Neural Networks*. [Online]  
Available at: <https://towardsdatascience.com/tagged/neural-networks>  
[Accessed 2 May 2018].

Trader, J., 2012. *Iris Recognition vs Retina Scanning What are the differences?*. [Online]  
Available at: <http://www.m2sys.com/blog/biometric-hardware/iris-recognition-vs-retina-scanning-what-are-the-differences/>  
[Accessed 15 November 2017].

Turgeman, A., 2017. Behavioral Biometrics Are Not New, So Why Are They So Hot Right Now?. *Forbes Tech*, 20 June, p. 2.

ujjwalkarn, 2016. *An Intuitive Explanation of Convolutional Neural Networks*. [Online]  
Available at: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>  
[Accessed 3 May 2018].

V.Chande, D. S., 2012. *A Survey on Generation of Test Cases and Test Data*, s.l.: Research Gate.

VV, P., 2016. *Mathematical Foundation for Activation Functions in Artificial Neural Networks*. [Online]  
Available at: <https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089>  
[Accessed 19 October 2017].

Walia, A. S., 2017. *Activation functions and it's types-Which is better?*. [Online]  
Available at: <https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>  
[Accessed 25 November 2017].

## Appendices

### Work through of an XOR in feed forward propagation

XOR Truth Table

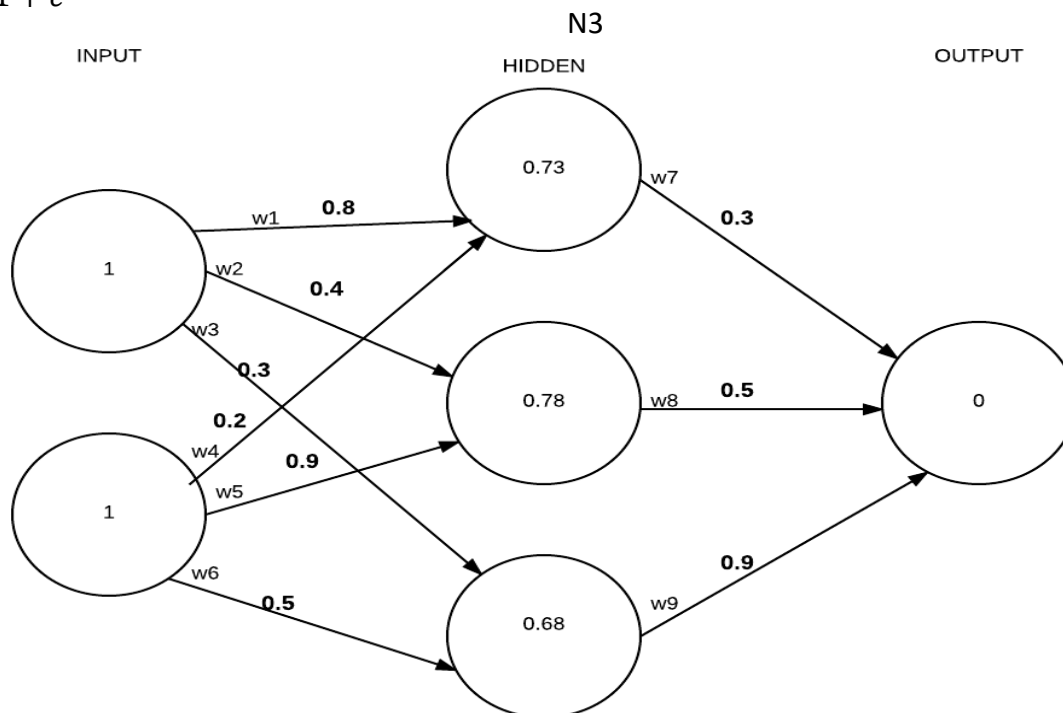
INPUTS		OUTPUTS
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

#### Steps

- Set of random input weights between 0 and 1
- Random weights between 0 and 1
- First values of the hidden layer = Sum(Inputs X Corresponding Weights)
- Sigmoid of values of hidden layer from above (See sigmoid function below)
- Output = Sum(Hidden layer results X Corresponding Weights)
- Sigmoid of final output
- Margin of Error = expected – calculated

#### Sigmoid Function

$$\frac{1}{1 + e^{-x}}$$



#### INPUT → HIDDEN

$$\text{N3: } \sum w1 \cdot w4 = (1 \times 0.8) + (1 \times 0.2) = 1.0$$

$$\text{N4: } \sum w2 \cdot w9 = (1 \times 0.4) + (1 \times 0.9) = 1.3$$

$$\text{N5: } \sum w3 \cdot w6 = (1 \times 0.3) + (1 \times 0.5) = 0.8$$

#### HIDDEN → OUTPUT

$$\text{N6: } \sum (\text{N3} \cdot w7) + \sum (\text{N4} \cdot w8) + \sum (\text{N5} \cdot w9)$$

$$= (1 \times 0.3) + (1.3 \times 0.5) + (0.8 \times 0.9)$$

$$= 0.2 + 0.39 + 0.612 = 1.236$$

<b>Sigmoid of input to hidden result</b> $S(1) = 0.73105857$ $S(1.3) = 0.79583498$ $S(0.8) = 0.68997448$	<b>Sigmoid of hidden to output</b> $S(1.236) = 0.774866989$
<b>Margin of Error:</b> $0 - 77 = (-0.77)$	

### Backpropagation work through for XOR

A learning algorithm used by multilayer neural nets based on minimising the mean or total squared error over several iterations. Backpropagation is a special form of the delta learning rule. Network has one input, one output and at least 1 hidden layer - mainly used for pattern association

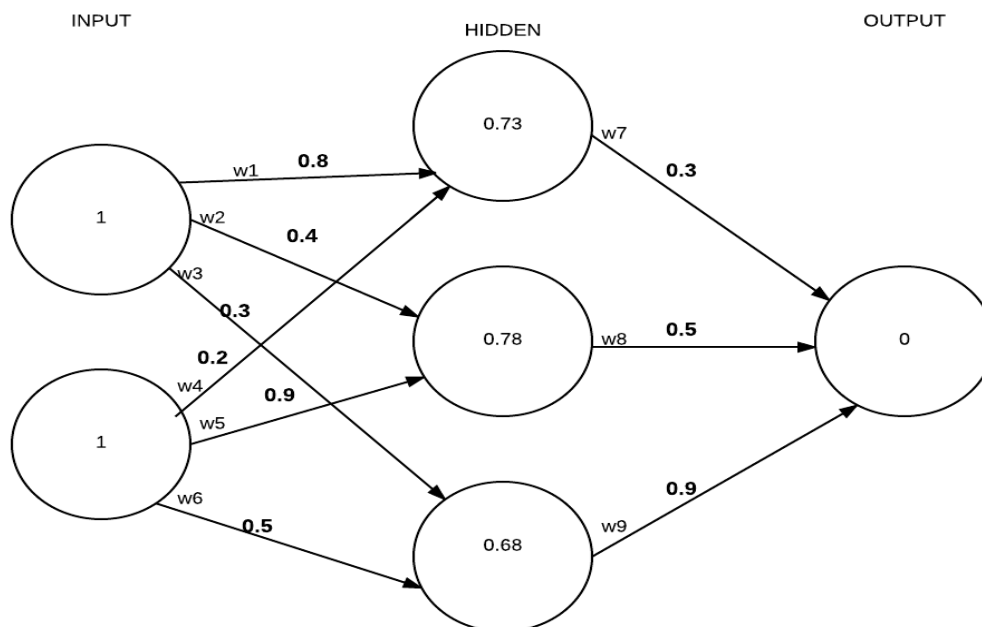


Figure 7.1 Feedforward work through of XOR ANN

### Sigmoid Function

$$\frac{1}{1 + e^{-x}}$$

<b>INPUT → HIDDEN</b>  $H1 = 0.73105857863$ $H2 = 0.78534983$ $H3 = 0.68997448$	<b>HIDDEN → OUTPUT</b>  $1.235$ Output layer MOE -0.77
---	---

Delta Output Sum =  $S(\text{Output}) \times \text{MOE (Margin of Error)}$

$$= S(1.235) \times (-0.77)$$

= -0.134398

### Output to Hidden

Delta Weights = Delta Output Sum x Hidden layer

-0.134398 . [0.731058, 0.78583983, 0.68997448] . [-0.09825, -0.10256, -0.092]

**W7** → 0.3 - 0.09825 = 0.202

**W8** → 0.5 - 0.10256 = 0.394

**W9** → 0.9 - 0.094 = 0.806

### Hidden to Input

Delta Hidden Sum = Delta Output Sum x Hidden To Outer Weights x S(HiddenSum)

-0.134 . [0.3, 0.5, 0.9] X S([1, 1.3, 0.8])

= [-0.0403, -0.0672, -0.1209] \* [0.1966, 0.1683, 0.2129]

= [-0.0079, -0.0113, -0.0259], [-0.0079, -0.0113, -0.0259]

**W1** → 0.8 - 0.079 = 0.7921

**W2** → 0.4 - 0.0113 = 0.3887

**W3** → 0.3 - 0.0259 = 0.2741

**W4** → 0.2 - 0.0079 = 0.1921

**W5** → 0.9 - 0.0113 = 0.8887

**W6** → 0.5 - 0.0259 = 0.4741

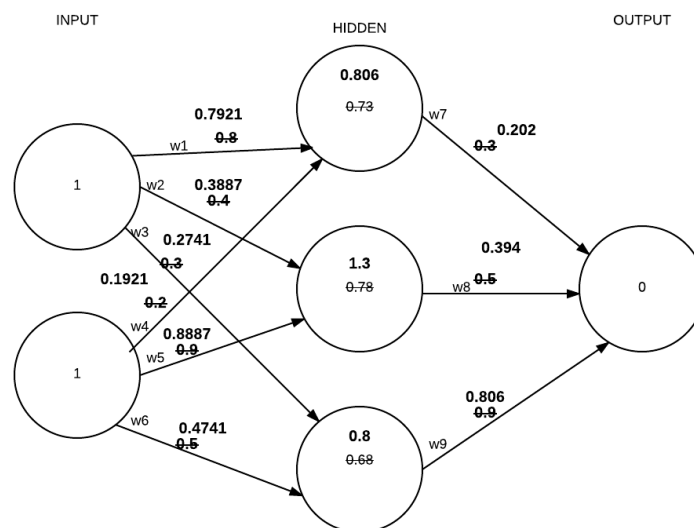


Figure 7.2 Readjusted weights following backpropagation of XOR ANN