University of Portsmouth

School of Engineering

Fingerprint Recognition System using Neural Network

Kafayat Adunola Adeoye

UP796445

BEng (Hons) in Electronic Systems Engineering (DL)

Supervisor

Dr Abdsamad Benkrid

May, 2017

Dedication

I dedicate this project to GOD almighty, my parents and sibling.

Acknowledgement

I would like to express my profound gratitude to all those that have made it possible to successfully complete this project, without their support I may not be able to accomplish this goal.

My sincere appreciation to my supervisor Dr Abdsamad Benkrid for his support although the duration of this project. I also would like to thank my parents, sibling and colleagues for their kind co-operation and encouragement towards the completion of this project.

Table of Contents

List of Figures

List of Tables

Abstract

This project presents a fingerprint recognition system using a neural network. To establish an objective assessment of the proposed neural network algorithm, fingerprint images from National institute of standards and technology (NIST) database were used. Image processing operations were carried out on the fingerprints prior to extracting the minutiae which are set as input into the network for verification or identification of a person. However, these processes are crucial to the performance of the neural network.

Back-propagation neural network algorithm called Scaled Conjugate Gradient is used to train the network. The aim of this project is to implement faster and reliable fingerprint minutiae matching algorithm and the Matlab experimental results show that the network has achieved an excellent performance in pattern recognition. Furthermore, the overall error rate is very minimal and the network generates 93.2% of accuracy for the fingerprint recognition system.

Chapter 1

Introduction

Fingerprints were used as signatures for commercial transactions and as a person's identity during the ancient time; this had led to the creation of fingerprint biometric systems in use today. In 1880, Francis Galton established the classification of the fingerprint which was later adopted by Edwin Henry in 1896, to develop a prototype fingerprints classification system using the classes of fingerprints for forensic investigation (Mohamed, et.al.,2012). The manual system created by Henry was time-consuming and cumbersome these have prompted law enforcement agencies like the Japanese National Police agency in 1980 to vastly research and design an automated fingerprint identification system (Maio, et.al.2009).

Fingerprint biometric technology has proffered a reliable solution as opposed to the use of conventional methods that use password or tokens to authenticate an individual. The aim of this project was to create a robust system to effectively extract the distinctive features of a fingerprint and the use of a neural network algorithm for recognition.

In the last few decades, the application of fingerprints recognition system has increased tremendously due to its convenient usage, cost-effectiveness and unique characteristics. It is becoming essential to find a viable matching algorithm. However, the artificial neural network provides the solution for a stable, accurate, minimal error, less time consuming and less sensitive to an environmental factor.

A neural network is defined by (Simon, 2015) "as the parallel distribution of simple processing units called neurons which has a natural propensity to store knowledge acquired through the learning process and making available for later use" the interconnection of these neurons is known as weights. The neural network performs the fingerprint pattern recognition by undergoing training of a set of input to be able to identify the pattern features from the information it has extracted (Simon, 2015). The report presents the Matlab implementation of a neural network as the matching algorithm, the image processing and feature extraction algorithms for the fingerprint recognition system.

1.1 Outlines

Chapter 1 gives an overview of the fingerprint recognition system using a neural network. Chapter 2 presents features and classes of the fingerprint, brief fingerprint history, biometric system and review of fingerprint classification algorithms. Chapter 3 briefly outlines the image processing, feature extraction and matching algorithms, along with the description of fingerprint image processing, which includes Median filtering, Contrast limited adaptive histogram equalization, Binarization and Thinning, and also describes the feature extraction algorithms using Local Binary Pattern and Gabor Filtering. Chapter 4 gives the overview of an artificial neural network, the backpropagation algorithm, including conjugate gradient algorithm and scaled conjugate gradient descent back-propagation algorithm Chapter 5 shows the analysis of the system implementation for fingerprint recognition. Chapter 6 presents the project conclusion and recommendation

Chapter 2

Fingerprint Biometrics

2.1 Introduction to Fingerprint Biometric

Biometric in Greek word literally means "life measurement". Biometrics is defined as the measurement of human characteristics known as the biometric identifier to distinguish a person. These identifiers are categorized into physiological characteristics, examples are fingerprint, face and iris and behavioural characteristics, examples are handwriting and voice ("Biometric", 2017).

Some key factors such as universality, uniqueness, permanence, acceptability and measurability are put into consideration to evaluate the relevant traits used in a biometric system. Fingerprint defined by (Boviks, 2009) as a smoothly flowing pattern formed by ridges and valleys as shown in figure 2.1. The pattern is formed from a natural secretion of the sweat gland in the epidemic layer that produces new skin cells within two months of pregnancy.

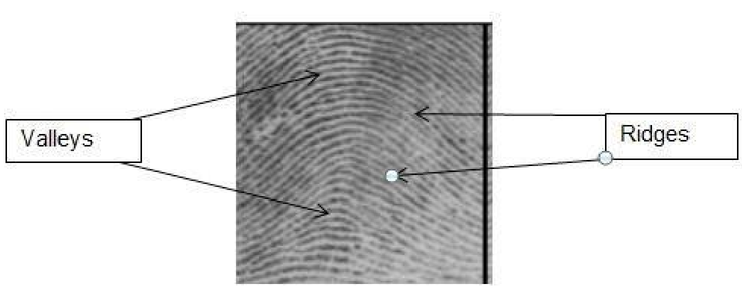

Figure 2.1: Fingerprint ridges and valleys

A fingerprint biometric system involves the automatic verification of fingerprint characteristics. It is used in several applications such as forensic investigation, control access, identification and immigration (El-Abed, et.al.2012). Fingerprint biometric system process involves three stages mainly; Enrolment, Verification and Identification as described in figure 2.2 and 2.3

- **Enrolment**: is the initial process of collecting raw fingerprint data sample from an individual and storing the captured image as a reference template in a database for matching.
- **Verification:** at this stage, the system executes a one to one matching of the captured biometrics and the database template. It is the process of providing a matching score between 0% and 100% and verifies if a person is who they claim to be.
- **Identification**: The system executes a one to many matching, to know "who a person is". That is recognizing an unknown or known biometric against a database. It is the process of either confirming or rejecting a person based on their physical features.

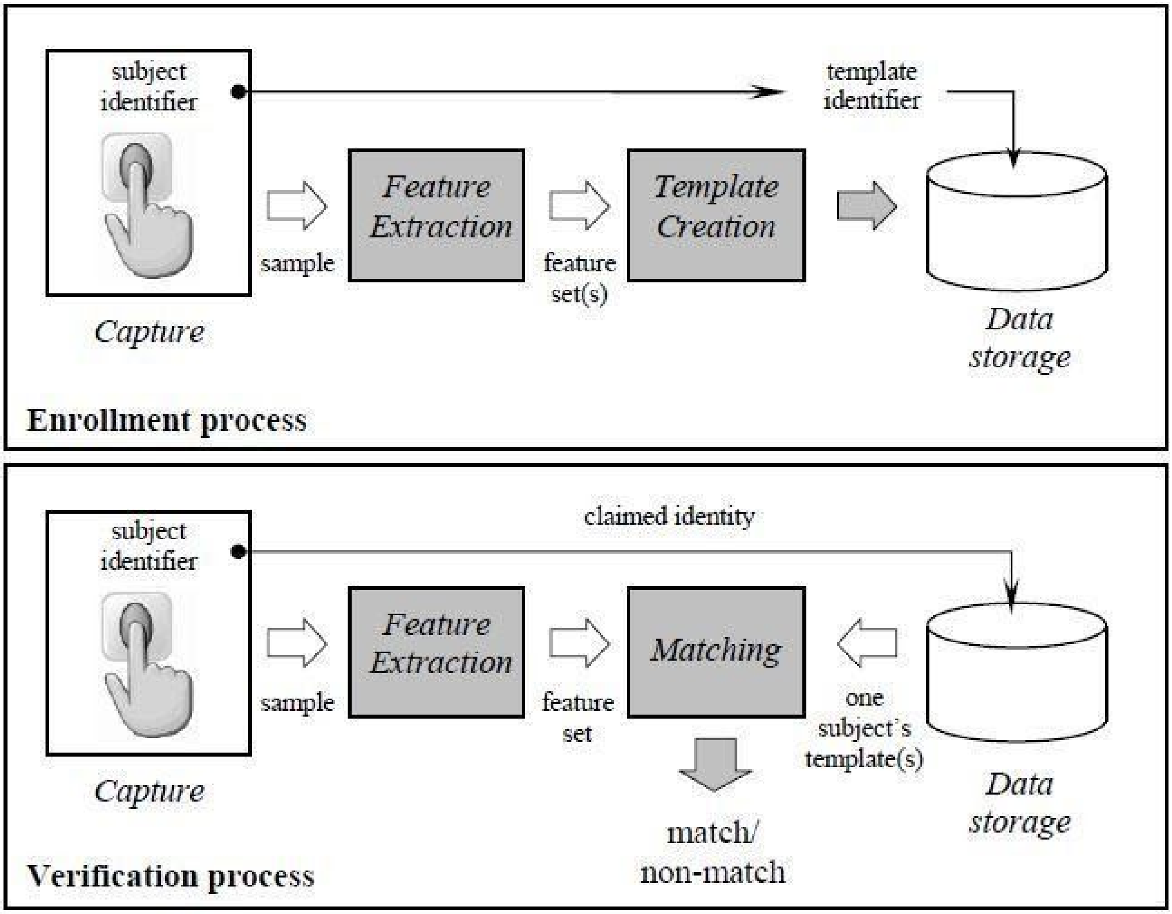

Figure 2.2: Enrolment and Verification system (Maio,et.al.2009)

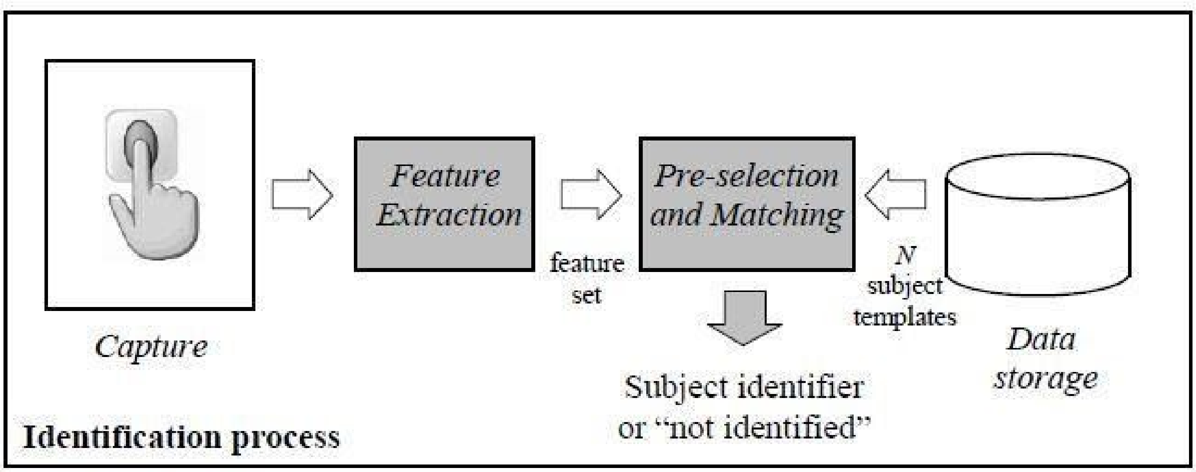

Figure 2.3: Identification system (Maio,et.al.2009)

2.1.1 Advantages and limitations of fingerprint biometric system

Fingerprint biometric system has its advantages and limitations; some are listed as follows,

Advantages

- The system is efficient and effective in usage
- Economical
- High accuracy
- Requires less storage
- It provides satisfactory security contrast to conventional methods of using password or token which can be easily forgotten.

Limitations

- The system is subjected to not been 100% accurate in performance due to error, system can be affected by environmental factors.
- Some people might find it intrusive
- Sensitive to the finger's skin dryness or dirty and may not suitable for children.

2.2 Fingerprint Features and Classification

Fingerprints pattern characteristics are majorly classified into:

- **Arches**: the print pattern flows upward and downward, it constitutes 5% of the population, and examples are plain arch and tented arch.
- **Loop**: the print pattern begins from one side of the finger, curvatures around to the other side, it constitutes 65% of the population, and examples are Left loop and Right loop.
- **Whorl**: the print pattern forms a circular or spiral shape, it constitutes 30% of the population, and examples are loop whorl, double loop whorl.

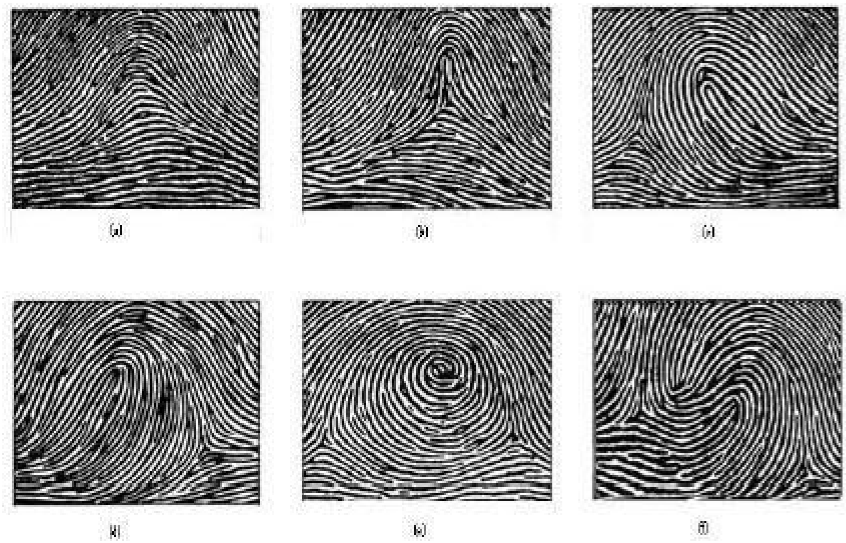

Figure 2.4: Classification of fingerprints a) Arch b) Tented Arch c) Right loop d) Left loop e) Whorl f) Double loop whorl (Navrit, Amit, 2011)

The ridge characteristics are the unique attributes of a fingerprint pattern called Minutiae. Minutiae are used in the matching process; there are four commonly used minutiae which are ridge termination, ridge bifurcation, delta, and core. Figure 2.5 shows an example of a delta and core points in a fingerprint.

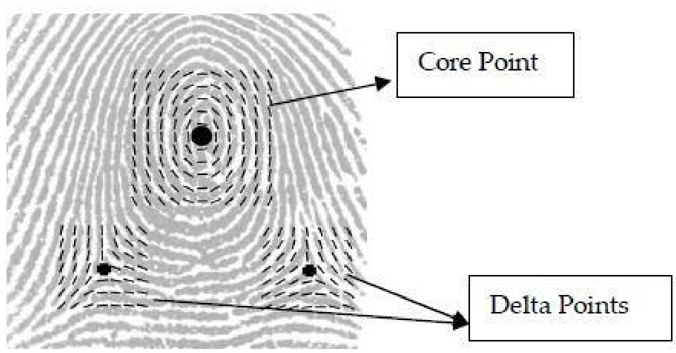

Figure 2.5: Fingerprint core and delta points (Navrit, Amit, 2011)

2.2.1 History of fingerprint

Fingerprint well-known as a reliable biometric characteristic has been in use since the BC, for transacting businesses in ancient Babylon and in China between 221-206 BC for forensic investigation, it gained popularity in the 14th century. Briefly outlined below are some of the scientist contributions to the discovery of a fingerprint ("History of fingerprint", 2007).

I. Mayer J.C.A (1788) a German anatomist found that fingerprints ridges are not same for two individuals
II. Hermann Weleker (1856) studied the permanence property of a fingerprint.
III. Thomas Taylor (1977) observed the application of fingerprint for criminality forensic
IV. Henry Faulds (1870) discovered that fingerprints can be used for a person's identification
V. Alphonso Bertillon (1882) implement the classification of fingerprints using anthropometry for police investigation
VI. Francis Galton (1880) identifies the classes of ridges characteristic as loop, whorl, tent and arch they are used till today for fingerprint classification.

2.3 Fingerprint Classification Algorithms

- **Rule-based algorithm:** this comprises a set of prediction model rules based on decision making such as the IF and then conditions and decision tree rules. Rule-based detect the numbers and points of the core and delta as described in figure 2.5 which are called singularity points to classify fingerprints (Alaa, Ghazali,2014). The algorithm is established using a mask to compute points and detect the singular points using Poincare index.

- **Syntactic (Structural) approach**: this algorithm is a tree-like structure of patterns, using grammars syntax language for classification. It has the ability to describe the ridge structure using a small set of pattern features to match reference features (Alaa, et.al.,2014). The approach involves the partitioning into the region the ridge flow as shown in figure 2.1 for classification using graphical relation.

- **Deep neural network (DNN):** these algorithms are more accurate in classification than the algorithms mentioned above. It uses sparse autoencoder, Back-propagation, Recurrent algorithm and so on to train for fingerprint features classification. DNN is a machine learning network using supervised and unsupervised learning algorithms (Ruxin, et.al.,2014). Figure 2.6 shows a simple neural network, that consist of an input layer(x), weight(y) and an output layer(z).

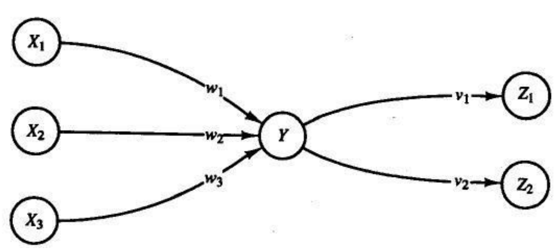

Figure 2.6: Simple neural network (Laurene, 1994)

Chapter 3

Fingerprint Recognition System Design

A fingerprint recognition system can either be implemented as a verification system or identification system depending on the required application. The system architecture is divided into four modules as shown in figure 3.1; each is discussed in this chapter as follows.

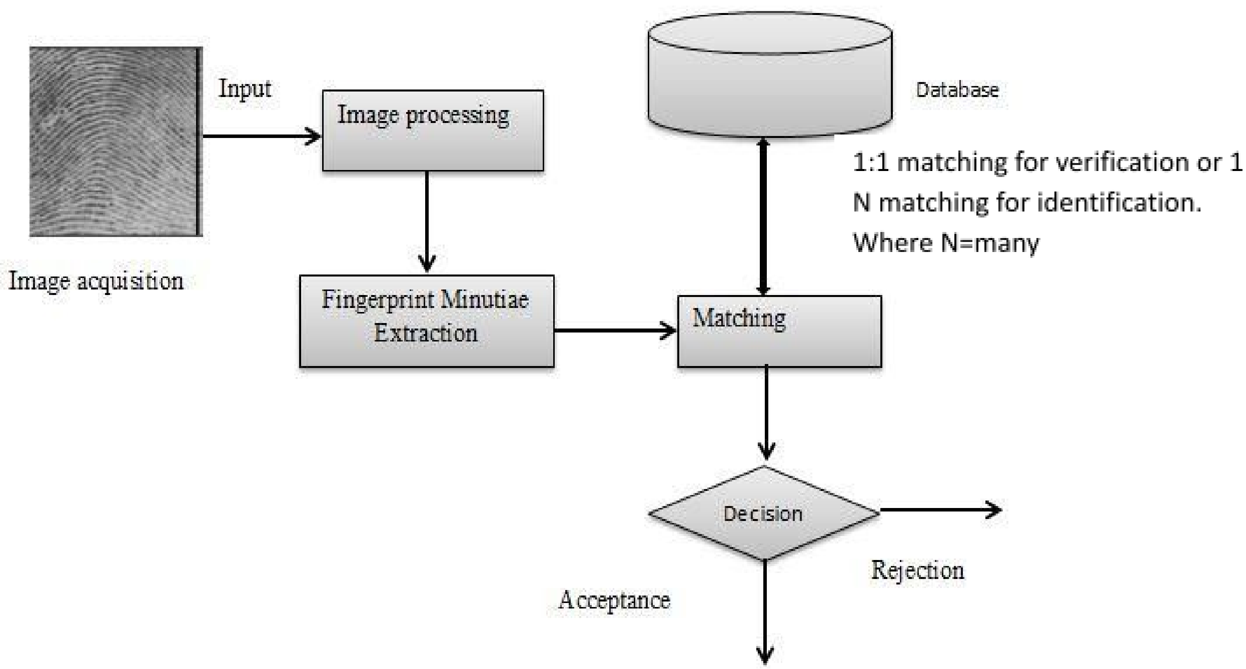

Figure 3.1: A typical fingerprint biometric system

3.1 Image Acquisition

This is the initial stage in the fingerprint recognition system, for capturing raw fingerprint data and then present as a digital image (El-Abed, et.al.2012). Raw fingerprints can be collected in two ways namely; ***offline mode and online mode.***

The *offline mode* is when ink is used to obtain fingerprint on a piece of paper, which is then transformed into a digital image. The *online mode* does not require the use of ink, instead, a fingerprint sensor which could either be a single fingerprint scanner or a multiple fingerprint scanner is used (Mohamed, Christophe,2012). The live scanner examples are ultrasound and optical scanner, figure3.2 shows some of the captured fingerprint grayscaled images of size 512 by 512-pixel resolutions obtained from National Institute of Standards and Technology website ("NIST",2017).

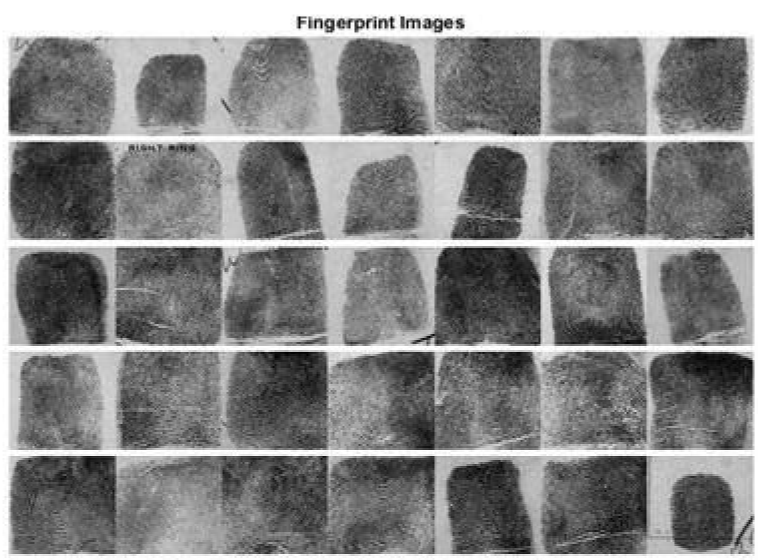

Figure 3.2: Captured fingerprint images

3.2 Image Processing

The purpose of this stage is to enhance the quality of fingerprint images in order to accurately extract the fingerprint minutiae. It includes noise filtering, image normalization, binarization and thinning (Ryu, Kong, & Kim, 2011). The matching performance could fail to detect authentic minutiae if the fingerprint image is distorted or of low quality. Therefore the following are some of the preprocessing phases;

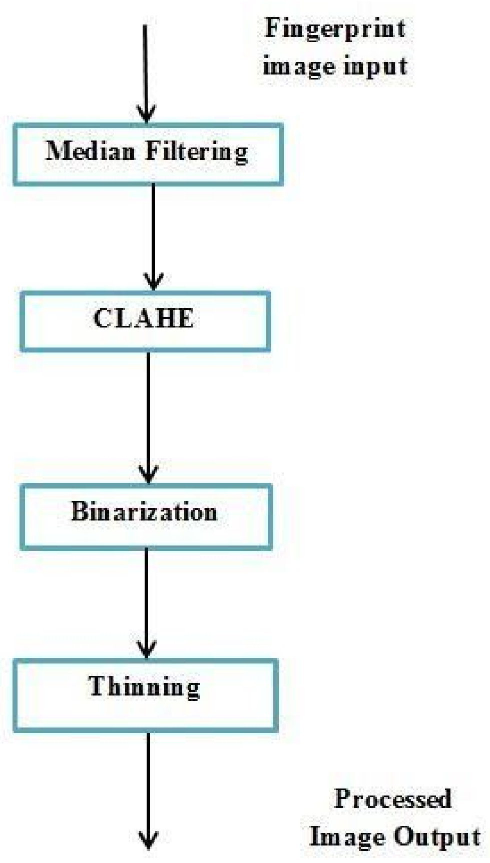

Figure 3.3: Fingerprint image pre-processing algorithm

3.2.1 Median filtering

A) Noise: While capturing raw fingerprint image from a scanner or when transmitting the image, a random variation called noise is introduced into the image intensity. Factors such as environmental condition and the quality of sensing element can affect the amount of noise generated in a digital image (Gonzalez,, Richard, 2008). The following are some of the common types of noise;

- Additive noise: it is introduced by sensors whereby the original image is corrupted with Gaussian noise.
- Multiplication noise: noise from imaging system like ultrasound, photographic plates
- Impulse noise: noise caused by electromagnetic interference and can easily be noticed in a digital image due to contrast distortion.
- Quantization noise: It is introduced by quantization in signal processing and telecommunication system. It is a signal-dependent noise that generates spurious content in a digital image. (Tinku, et.al., 2005).

B) Median Filtering

In order to reduce noise degradation from fingerprint images, the median filtering operation is used. The median filter is a nonlinear filter applied to a grayscaled image using a window size of m by n over the neighbourhood pixels to smoothen and remove noise (Gonzalez, Richard, Steven, 2016).

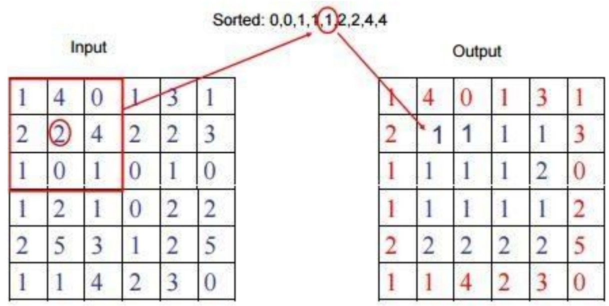

Figure 3.4: Example of a 2D median filtering using 3 by3 windows ("Image Filtering",2010).

It is an example of a spatial filter also known as order filter which output depends on the ranking of pixels in an image neighbourhood and return with the centre pixel value, in the case of median filtering it computes the median pixel values from the ranking outcome (Gonzalez,, Richard, 2008). For example, in a 3 by 3 neighbour as shown in figure 3.4 the 5th value is the median. However, this filter does not shift the boundaries of the pixels, it perverse the feature of the image, capable of better noise reduction with less blurring and are less sensitive to outliers. (Tinku, Ajoy, 2005).

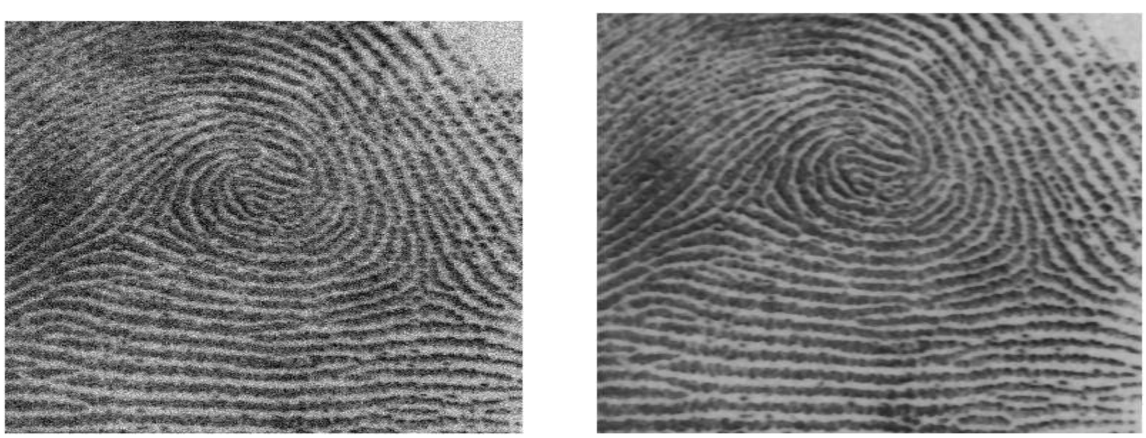

Figure 3.5: Noisy fingerprint image and Median filtered fingerprint image

3.2.2 Normalization

Normalization is also known as contrast stretching or histogram stretching in image processing, is the alteration in the range of pixels intensity values in a digital image ("Normalization" n.d). Poor illumination of an image indicates low contrast; this can result from insufficient range from the imaging sensor during acquisition (Gonzalez,, Richard, 2008). For instance the uses of contract limit histogram equalization and histogram equalization to increase the grey level dynamic range in an image.

A) Histogram Equalization

The histogram represents the frequency of an image grey level. If an image is poorly visible the histogram will be narrow and centred towards the middle scale, while a widely distributed histogram as shown in figure 3.6 reflect a high contrast image with all grey level present in the image (Gonzalez, et, al,.2016) It can be used to determine the condition of an image.

Histogram equalization (Gonzalez, et, al,.2016) transforms the intensity value of a digital image so that the histogram output image matches a specified histogram. Histogram equalization can over amplify noise and generates a poor image as many pixels have the same grey level. However, to avoid this image saturation contrast limit adaptive histogram equalization with clip limit is recommended. (Sepasian et al.,2008)

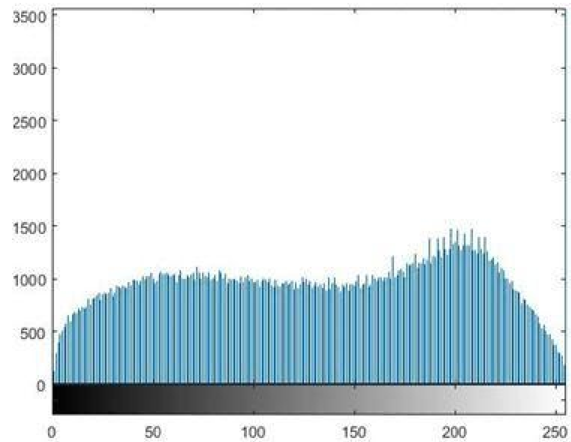

Figure 3.6: High contrast fingerprint image histogram

B) Contrast Limit Adaptive Histogram Equalization (CLAHE)

CLAHE is the process of enhancing the contrast of a grayscaled image and it operates on small regions called “tile”, each tiles contrast is enhanced on the image (Gonzalez, et, al,.2016). CLAHE combines the neighbour tiles using bilinear interpolation and removes artificially induced boundaries in the image to derive a uniformly distributed intensity level. (Bovik,2009). The following are the set CLAHE parameters;

- CLAHE clip limit parameter is a contrast factor that avoids saturation of images in the homogeneous area in order to reduce false fingerprint minutiae. The clip limit value is specified as a real scalar in the range [0, 1]. Higher limits generate more contrast (Gonzalez, et, al,.2016), without this limit CLAHE can generate an image that not as good as the original image.
- The number of tiles: this depends on the type of image; it is determined by a two-element vector which divides the original image into columns and rows of tiles. (Tinku,el,at.,2005)

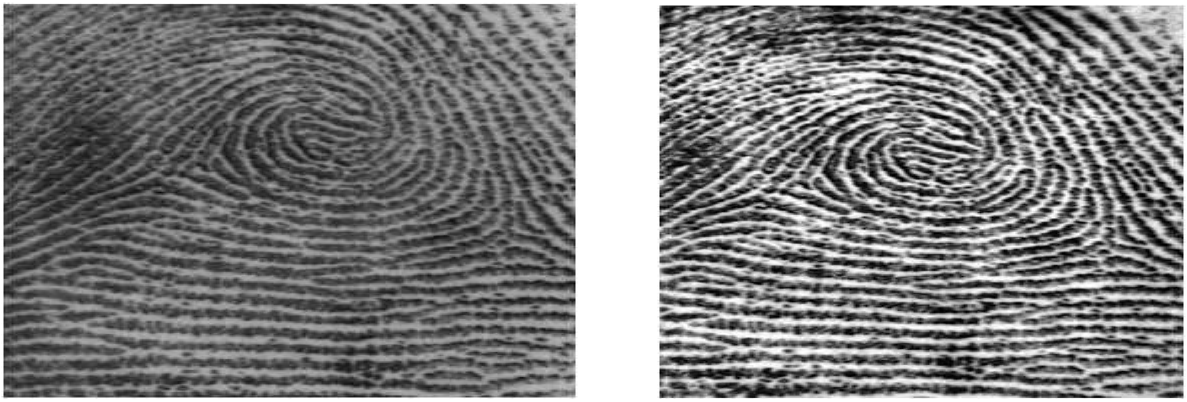

Figure 3.7: Median filtered and CLAHE processed fingerprint image

3.2.3 Binarization and Thinning

Binarization is the conversion of a grayscaled image into a binary image also known as monochrome. It is the process of replacing each image pixel with 1s and 0s using the threshold method (Puneet, Naresh,2013). The threshold is the process of comparing each pixel value in an image to the grayscaled range values. Its operation is given as follow;

If pixel< threshold value

Then new pixel =0 otherwise 1

Binarization algorithm can be grouped into Global and Local method. A global method is when the set threshold value is used on the whole image; examples are the Otsu method and kitler method (Puneet, Naresh, 2013). While the local method such as the one proposed is when the set threshold value is used on the image pixel by pixel; examples are Adaptive method and Niblack method (Puneet, Naresh, 2013). However, binarization is an important step for preprocessing image of low resolution and to segment the image foreground from the background (Parker,2011). Figure 3.8 shows an example of a binarized digital image.

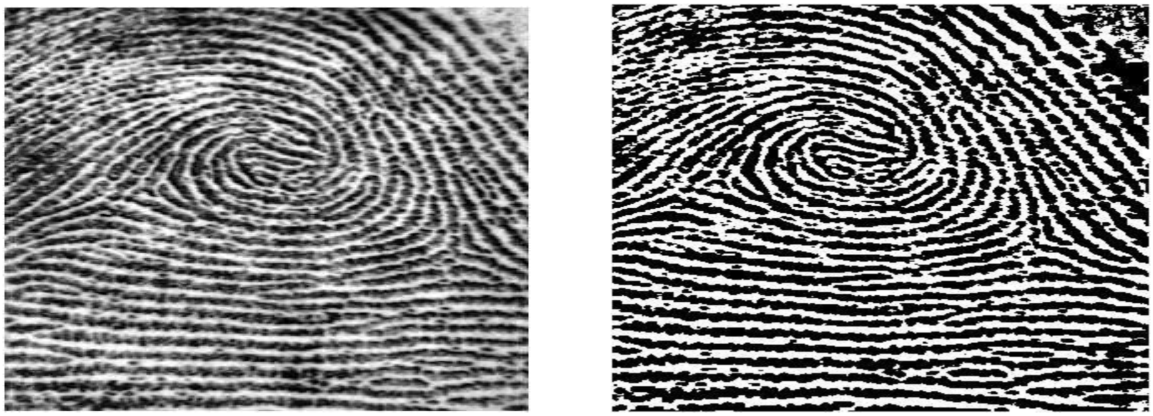

Figure 3.8: CLAHE fingerprint image and Binarized fingerprint Image

Thinning: is an example of morphological operation performing on binary images only (Parker, 2011). It generates a skeleton-like image in relation to the number pixels and the algorithm repeatedly reduces the pixel layers till it is of a single-pixel wide (Gonzalez, Richard, 2008).

Thinning however sometimes generate unwanted spurs also known as parasitic components in an image. This can be removed by a method called pruning that iteratively recognize and remove spurs, it acts as a post-processing operation in thinning (Gonzalez, et, al,.2016). Figure 3.9 shows a thinned fingerprint image, thinning has no effect on the topological structure of the fingerprint image.

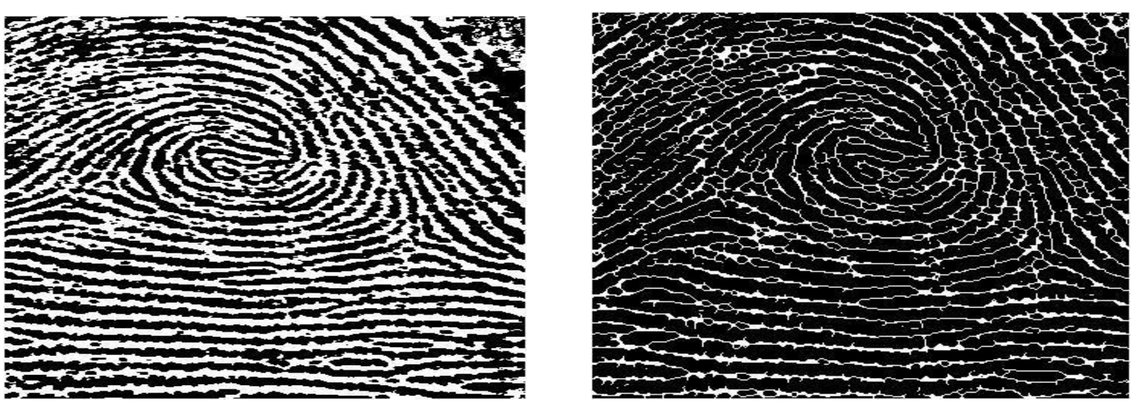

Figure 3.9: Binarized fingerprint image and Thinned pre-processed fingerprint Image

3.3 Minutiae Extraction

This is the process of extracting minutiae from fingerprint images for recognition. A good extraction algorithm must be able to retain most of the fingerprint image minutiae after extraction (El-Abed, et.al.2012). Gabor filtering is often used to extract the features while the Local Binary Pattern extracts more finely detailed features from the Gabor response and reduces the vector dimension for a faster recognition process. (Di,et.al.,nd)

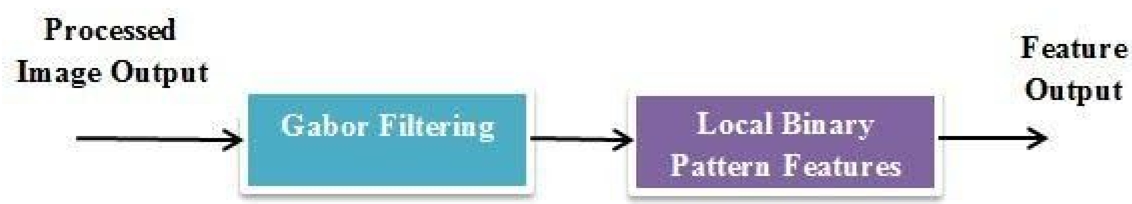

Figure 3.10: Feature extraction algorithm flowchart

3.3.1 Gabor filtering

In 1946, Dennis Gabor introduced the convolution of two functions to represent a signal. Gabor filtering functions in the frequency or spatial domain and acts as a bandpass filter. Gabor filtering has been used in several applications such as texture analysis, feature extraction, discrimination and so on ("Gabor filtering", 2017). A 2D Gabor filter has a set of frequencies and orientations which are useful for the extraction of image feature resulting in 2D Gabor response magnitude for pattern recognition (Tinku,et.al.,2005). The extracted feature response is the convolution of the image with the Gabor filter bank ("Gabor filtering", 2017).

Gabor filter response consists of real and imaginary components as shown in figure 3.11 a) and b) formed into a complex number. The general form of a Gabor filter response in the x and y-axis of the sinusoidal plane is defined as described in equation 3.1(Ilonene, Kamarainen, Kalviainen, 2005)

$$G_{(x,y)} = \frac{1}{2\pi\delta_x\delta_y} \exp\left[-\frac{1}{2}\left(\frac{x^2}{\delta_x{}^2} + \frac{y^2}{\delta_y{}^2}\right)\right] \exp j(2\pi\mu_o x) \ldots\ldots\ldots\ldots\ldots\ldots(3.1)$$

Where frequency sinusoidal carrier in the cartesian coordinate is μ_o, δ_x *and* δ_y are the constant values that define the Gaussian envelope. Furthermore, one main advantage of Gabor is that it minimizes random noise and smoothen irregularities in the image structure. Figure 3.11 demonstrate an example of a 2 dimensional Gabor filter response with a set of frequency and orientation.

Moreover, Gabor filtering has the following properties; 1) Frequency and orientation are tunable parameters for feature extraction, 2) If the constant value is small then the Gabor response at different orientations will be small.

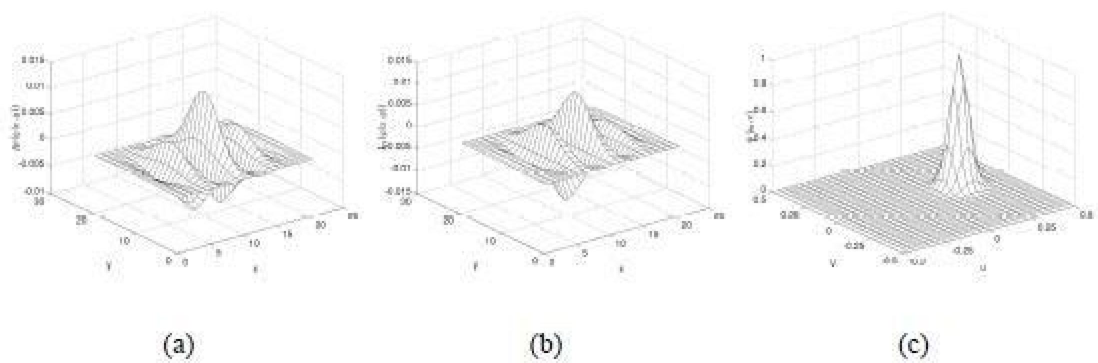

Figure 3.11: Example of a 2D Gabor filter with frequency=0.2, orientation=0^0, a) Magnitude; b) Phase c) Frequency domain (Ilonene, et.al.,2005)

A) Gabor filter bank

Gabor filter bank contains frequencies and orientations parameters that are adjustable in order to configure the filter (Gonzalez, et, al,.2016). Gabor bank frequencies and orientations are generated with Matlab codes presented chapter 5 sections 5.3.1, Gabor filter bank of frequency [6] and orientation [90^0] is used to extract minutiae from the input fingerprint images vectors.

- **Frequency** is the sinusoidal carrier in the image pixel to determine the cut off of the filter response. The frequencies of the bank are defined by

$$f_o = f_{max} \ , f_1 = \frac{f_{max}}{k} \ , f_2 = \frac{f_{max}}{k^2} \ \ldots\ldots\ldots\ldots\ldots f_n = \frac{f_{max}}{k^{m-1}} \quad \ldots\ldots 5.2$$

  Where f_o represents the filter frequencies scaling factor, f_{max} is the maximum tuned frequency and k is a constant value.

- **Orientation** of the filtering measured in degree is the direction of the sinusoidal plane wave (Ilonene, et.al.,2005). In figure 3.12 different orientations of a 2D Gabor filter in the frequency, space is shown

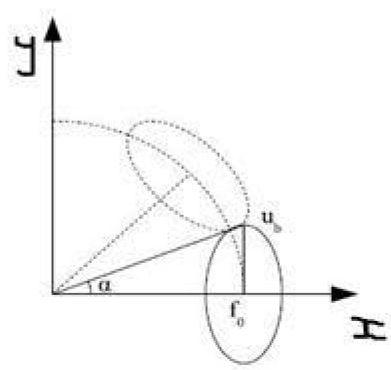

Figure 3.12: 2D Gabor filter with different orientations (Ilonene, et.al.,2005)

3.2.2 Local binary pattern (LBP) feature

Local Binary Pattern in computer vision is used as a visual descriptor for classification. Its extracts feature from images; this is done using circular neighbourhoods threshold (as in figure 3.13 which are of various types) multiplied with image pixel. The feature vector dimension is reduced to achieve robustness, speed and real-time optimization (“Local binary patter”,2017).

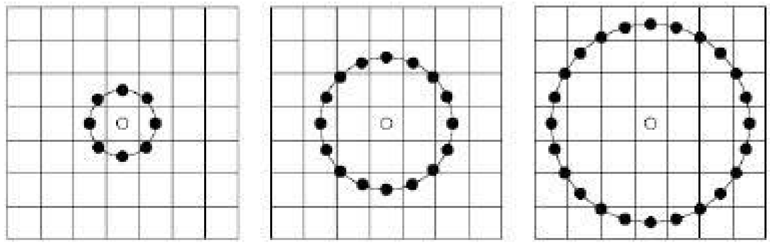

Figure 3.13: Example of a circular LBP operator (8,1),(16,2) and (24, 3) neighborhoods (Di, et.al.,n.d).

LBP operator labels the image pixels with binary numbers as shown in figure 3.14, the resultant is called LBP which encodes the image pattern. (Di, H., Caifeng, S., Mohsen, A., Yunhong, W., Liming,n.d). The LBP operator decreases the number of neighbour pixels in an image or selects a subset of the image histogram bins as a means of reducing the dimension (Pietikainen, Hadid, Zhao, Ahonen, 2011). LBP operator is invariant to monotonic grey scale transformations to preserve image pixel intensity order in the neighbourhood (Di, et.al.,n.d).

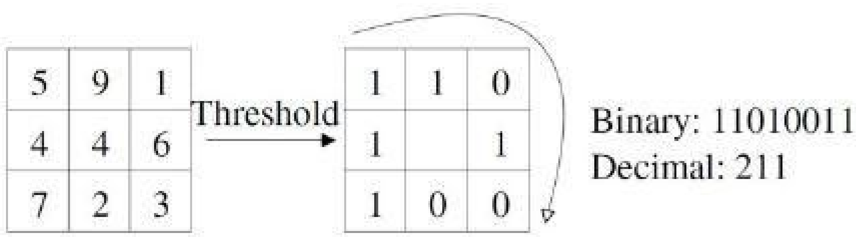

Figure 3.14: Example of LBP operator (Di, et.al.,n.d)

Moreover, one factor that affects a neural network computation is the length of the feature vector for this research LBP reduces the fingerprint minutiae vector dimension of the Gabor filter to 59 and implement a simple rotation-invariant descriptor. Other extensions of Local binary pattern includes, transition LBP, multi-block LBP, modified LBP and direction coded LBP ("Local binary pattern" 2017). The following are the parameters set for Local binary pattern feature;

i. The number of neighbours: is used to compute the LBP for each pixel in the input vector with positive integers.

ii. Circular: forms a circularly symmetric pattern around each pixel and the required radius is selected for the circular pattern.
iii. Rotation invariance: it is desirable to have features robust to rotations of the input image; therefore this parameter can be set.
iv. Histogram: determines the distribution of the binary pattern which is uniform in this case.
v. Cell size: are set to moderately extract information over a large region. If cell size is too large image feature details could be lost (Gonzalez, et, al,.2016)

The local binary feature vector is created with the following algorithm; ("Local binary pattern" 2017)

Step 1 each image pixel is split into cells using the LBP operator

Step 2 compared each cell of its neighbour in a circular manner

Step 3 where the centre pixel value is < neighbour value;

return 0 value, otherwise return 1 value; // this is later converted into decimal

end

3.4 Fingerprint Matching

Matching is based on fingerprint ridge pattern, however, it defines the similarity between the captured fingerprints and the database template in order to reach a decision to either accept or reject a person for verification or identification.

The extracted numerical minutiae vectors from the preprocessed images are the input into the multilayer neural network. This network is trained with back-propagation algorithm (Howard, Mark, 2002).

3.5 Database

The database is used as storage for the template created. It contains rows, tables and columns for collecting the fingerprint images prior to the preprocessing operation, feature extraction and matching (El-Abed, et.al.2012). Database management software such as MYSQL, MS SQL Server is the interactive interface between a user and the stored raw fingerprint images (template).

Fingerprint images stored in National institute of standards and technology (NIST) database is used for the project research.

3.6 Implementation Environment

Matlab 2017a trial version is used to carry out the implementation of the fingerprint recognition system. The Matlab software is integrated with Image processing toolbox used for Gabor filtering, Median filtering and Contract Adaptive Histogram Equalization, Computer Vision System toolbox used for Local Binary pattern feature extraction and the Neural Network Toolbox for matching or classification.

There are two ways to implement the pattern recognition network in Matlab. To either use the graphical user interface (GUI) nprtool or the Matlab command line. For the neural network matching algorithm implementation, the GUI was used and the command line functions were deployed from the GUI script interface which contains the Matlab codes for the network simulation process, this codes can be re-run or customized in the Matlab command line.

The hardware used to run the implementation is a 64-bit operating system Hewlett-Packard personal laptop with 2GB RAM and an AMD processor.

Chapter 4

Artificial Neural Network Matching Algorithm

4.1 Artificial Neural Network Overview

An artificial neural network is the computational model used in machine learning having similar characteristics with the biological nervous system, that constitutes dendrite which receives stimuli from the external environment through the neurons, soma to retain information received and an axon that act as the transmitting medium (Laurene, 1994). A neural network is the collection of information processing elements called neurons connected with silicon or wire to respond to external inputs ("Artificial intelligent- neural network", n.d).

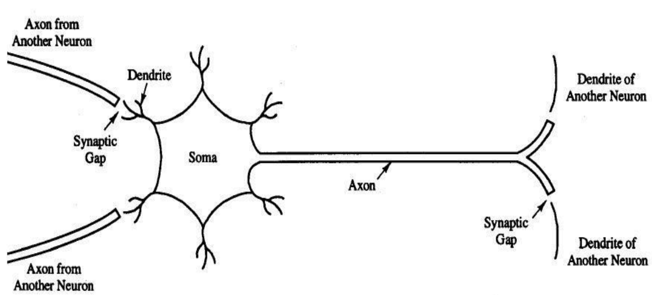

Figure 4.1: Example of a biological neuron (Laurene, 1994).

The artificial neural network comprises of its architecture, training algorithm and its activation function. The signal is passed between the neurons and each neuron has weights that are able to learn by altering its value (Laurene, 1994). Neural networks are used in several applications such as Robotic, Telecommunication, Vision and Control system and Pattern Recognition (Howard., Mark,2002).

There are two types of artificial neural network based on their functions;

- Feedforward artificial neural network: involving the unidirectional flow of information with fixed input and output. Single-layer and multi-layer neural network are examples.
- Feedback artificial neural network: consists of the feedback loop to the input ("Artificial intelligent- neural network", n.d).

Why use artificial neural network;

An artificial neural network is used because of its ability to adapt to a given task according to the data input for training; self-organized, is capable of organizing learned information, real-time computation, fast matching for pattern recognition and fault tolerance (Laurene, 1994).

4.1.1 Artificial neural network Layer

An artificial neural network is divided into layers as presented in figure 4.2; a layer is a vector of neurons which involves the combination of weights, multiplication operation, summing operation, biases and transfer function (Ivan, Danilo, Rogerio, Luisa, Silas,2017).

- **Input layer**: is where external data are inputted into the network.
- **Hidden layer** is where the internal activities of the network are activated and it consists of neurons that extract from the input data the features to be processed.
- **Output layer**: this layer output the result from the network hidden layer process (Ivan, et.al.,2017).

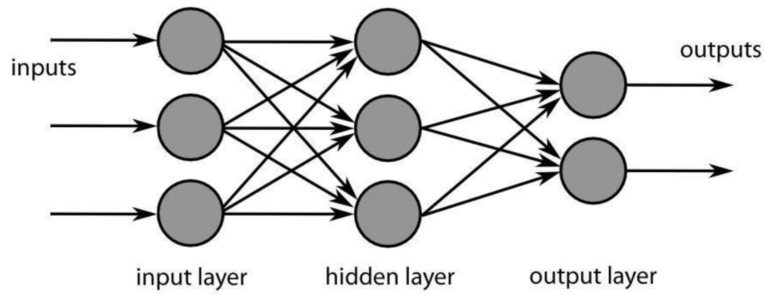

Figure 4.2: Example of an artificial neural network consisting of layers

4.1.2 Neurons

A simple neuron model connection multiplies it weight with input and sum with the bias as argument of the transfer function to generate a scalar output (Martin, et.al.,1995). The weights (w) and biases (b) are the neurons tunable parameters. Figure 4.3, describes a simple neuron model.

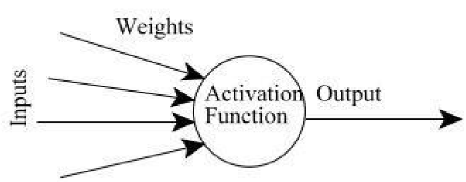

Figure 4.3: A simple neuron

Transfer function known as the activation function defines the output value for each target input. There are three commonly used transfer functions in a neural network, figure 4.4 shows the symbols of each transfer functions respectively (Martin, et.al.,1995);

i. Threshold transfer function: this set neuron output to 0 or 1. If a function argument is ≤ or ≥ the threshold value.

ii. Linear transfer function: this is used in an adaptive linear network, it takes input that ranges from plus or minus infinity.

iii. Sigmoid transfer function: present in the multilayer feed-forward network and the output value is between plus or minus infinity which varies continuously and not linearly with input change (Laurene, 1994).

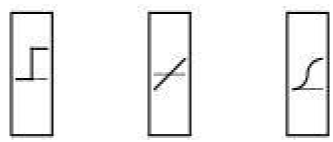

Figure 4.4: Symbol for each of the transfer functions (Howard, et.al, 2002)

4.1.3 Artificial neural network architecture

Artificial neural network architecture can include one or multiple layers consisting of neurons and the connection pattern. Consequently, the artificial neural network architecture is classified into three in terms of layer (Ivan, et.al.,2017);

i) **Single-layer network**: in this network, information flows in one direction. It contains one input layer and one output layer containing neurons, as described in figure 4.5 (Ivan, et.al.,2017). Examples of single-layer network are Perceptron network using Hebb's learning algorithm and Adaline network (Adaptive linear network) using delta learning algorithms (Martins, et.al.,1995).

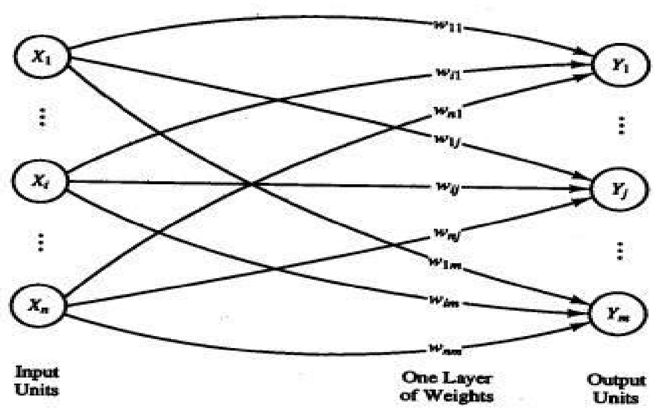

Figure 4.5: Example of a single layer network (Laurene, 1994)

ii) **Multilayer network**: this network consists of an input layer, hidden layers and an output layer and has more than one layers combined. (Ivan,et.al.,2017) Examples includes multilayer perceptron network and radial basis network, using delta learning and widrow hoff (back-propagation) algorithm. (Martins, et.al.,1995).

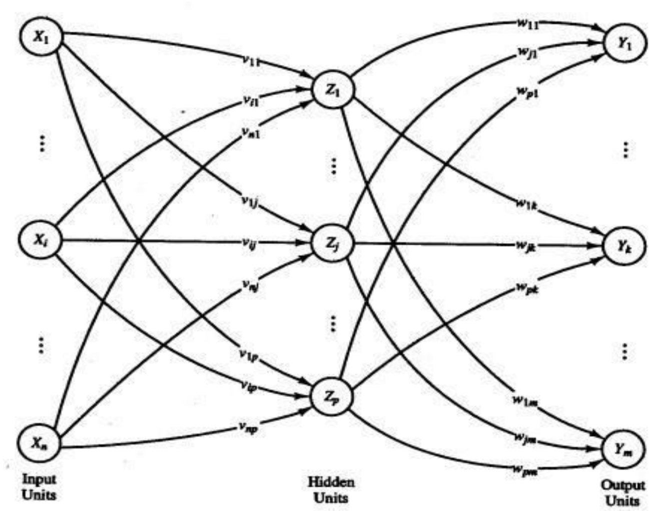

Figure 4.6: Example of a multilayer network (Laurene, 1994)

iii) **Recurrent network**: this network is called the feedback network, where the output from network neurons is fed back as input for other neurons (Ivan, et.al.,2017). It is used mostly in dynamic and control time-invariant system. Examples of this network include Hopfield and multilayer perceptron with feedback network, their learning algorithm is based on generalized delta rules (Martins, et.al.,1995).

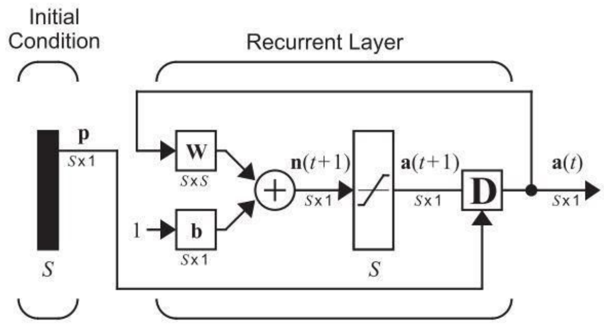

Figure 4.7: Example of a recurrent network (Martins, et.al.,1995)

4.1.4 Learning or training process

The training process is the procedures whereby the network Weights and Biases are adjusted, in order to execute a specific function this is also known as a generalization (Martins, et.al.,1995). Hence, the process is grouped into supervised learning, unsupervised learning and reinforcement learning (Ivan, et.al.,2017);

i. **Supervised learning**: is the continuous tuning of the weights and biases of the network neuron, by comparing the output with the target input. Figure 4.8 demonstrates the flow of supervised learning.

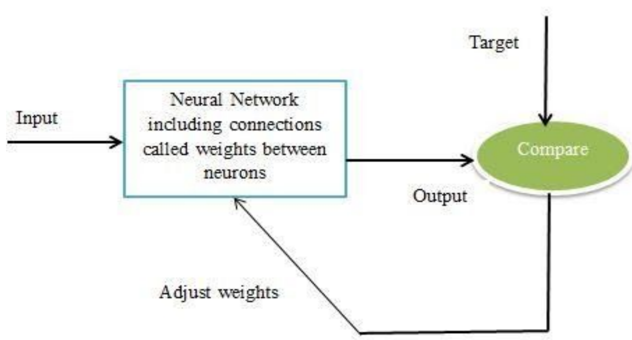

Figure 4.8: A supervised learning neural network flowchart

ii. **Reinforcement learning**: this learning shares similar characteristics with supervised learning, instead of making comparison; it measures the quality of the network performance to its inputs.

iii. **Unsupervised learning**: It involves altering the weights and biases in accordance to the network inputs only without a targeted output. It's used for vector quantization. (Ivan,et.al.,2017)

4.2 Back-propagation Algorithm

A multilayer feed-forward network is trained using the backpropagation algorithm to solve a specific problem. Back-propagation involves the process whereby the network first-order derivatives error are calculated in accordance with the network weights and biases to implement gradient descent which is an optimization technique (Simon, 2005). The algorithm is an example of supervised learning, to train the network to correctly respond to input using a sigmoid activation function. Backpropagation training begins with randomly altering weights and biases to minimize error (Christopher,1995).

(Christopher,1995) defines the mathematical model of the backpropagation algorithm which involves three main processes as commented in the algorithm steps below, (Laurene, 1994) describe the backpropagation algorithm as follow,

Step 1 initializes Weights and Biases

Step 2 for each training pair do step 3 and 7

// feed forward of input pattern

Step 3 signal received by each input units is forwarded to the hidden layer

Step 4 the hidden layer sum its weights and apply to the activation function to calculate the output and send to the output layer

// calculate the back propagation of associated error

Step 5 target input corresponding to the trained input is received at the output layer and compute the error by calculating the difference.

Step 6 calculate weights and biases

// updating the Weights and Biases

Step 7 each output layer and hidden layer updates it weights and biases

Step 8 check training stopping criteria

end

4.2.1 Backpropagation algorithms drawbacks

With adequate numbers of neurons in the hidden layer, back-propagation can perform any approximation function; in general, the specific number to appropriate cannot be determined. (Martins, et.al., 1995)

Still, it is observed that backpropagation cannot guarantee an optimum solution that is why it is necessary to retrain severally and reinitialize the backpropagation algorithm to memorize training with minimal error as it does not quickly generalize to new conditions. (Simon, 2005) However, it is essential that a network is able to successfully generalize what it has learnt by having few parameters than the training dataset. In addition, when the learning rate value is too high the network becomes unstable.

4.3 Conjugate Gradient Algorithm

The conjugate gradient is a general optimization method that employs search along the conjugate direction and step size (weight update) using second-order derivative information which produces a faster convergence (Moller, 1993). Backpropagation algorithm instead computes the first-order derivative of the direction in which its performance can decrease rapidly and this will slow the algorithm convergence (Martins, et.al., 1995). The commonly used types of line search function are as follow; (Howard, et.al.,2002)

i. Golden section search: this search is linear and does not calculate the slope. The rate of convergence begins when the algorithm has been initialized.
ii. Brent's search is a linear search that integrates golden section search and quadratic interpolation.
iii. Hybrid bisection cubic search: combine bisection and cubic interpolation.
iv. Charalambous search: a hybrid search that uses cubic interpolation and it is the default search line for conjugate gradient algorithms.
v. Back tricking search: used in quasi newton algorithm, while searching the step size the multiplier backtrack until it reaches a suitable performance.(Martins, et.al., 1995)

(Christopher,1995) describes the conjugate gradient algorithm as follows;

Step 1 select initial weight vectors

Step 2 evaluate the gradient and set the initial search direction to minimize error

Step 3 check that the training stopping condition is satisfied

Step 4 evaluate the new gradient vector

Step 5 evaluate new search direction

Step 6 set iteration (k) = k+1 and go to step 3; until stopping condition

end;

The conjugate gradient algorithm has four types (Howard, et.al.,2002)

- Fletcher Reeve update
- Polak Ribiere update
- Powell Beale Restarts
- Scaled Conjugate Gradient algorithm

4.4 Scaled Conjugate Gradient Algorithm

The first three conjugate gradient algorithms mentioned in section 4.3 make use of line search (Howard, et.al.,2002) which is computationally expensive. Line search requires computing training input for each search several times (Moller,1993). However, the scaled conjugate gradient algorithm proposed by (Moller,1993) was designed to use step size scaling and eliminate the time consuming line search procedures (Howard, et.al., 2002). (Christopher,1995) has mathematically illustrated the Scaled Conjugate Gradient algorithm by (Moller,1993) which is described below;

Step 1 choose initial weight vectors

Step 2 calculate second order derivative

Step 3 calculate scaled step size

Step 4 update weight vectors

If iteration (j) = 0 then restart algorithm;

else create new conjugate direction;

If the gradient descent direction is not equal 0;

then set iteration (j) = j+1 and go to step 2 else

end;

Chapter 5

Matlab Implementation of Fingerprint Recognition System Using Neural Network

A database template of 160 fingerprint images of 8 bits gray scaled level each of size 512 by 512 pixel resolution saved in portable network graphic (PNG) format was created from NIST fingerprints database. The database template consists of two pairs of fingerprints from 80 people of different fingerprint classes which is divided into input dataset and a target dataset to train the artificial neural network for the matching phase in the fingerprint recognition system. The extracted minutiae of the target dataset now serve as a benchmark for the network output to tell the network how to recognize the fingerprint pattern.

Therefore, a fingerprint image was used to test the network recognition for identification. The input dataset and target dataset fed into the network have to be preprocessed and the features extracted prior to training the network otherwise the network performance can be affected. The network after training matches any input fingerprint in accordance with what it has learned without necessarily matching against the database template for identification or verification of an individual.

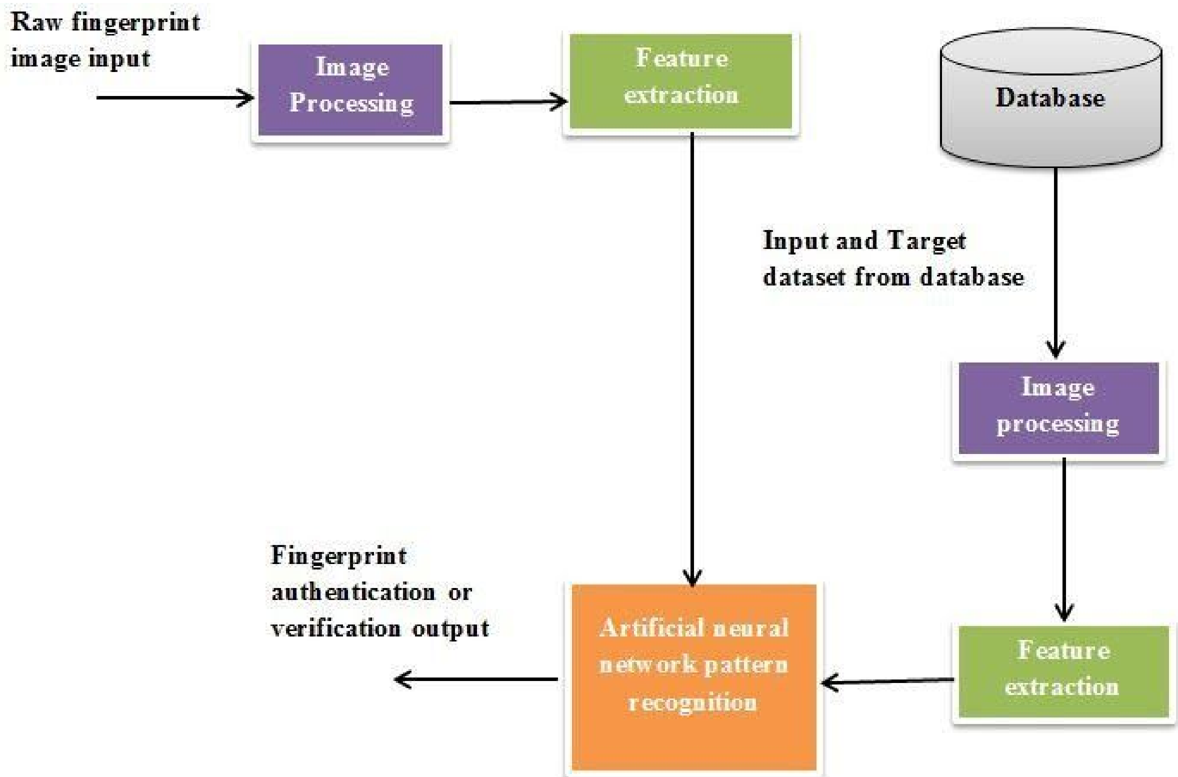

Figure 5.1: Implemented a fingerprint recognition system using a neural network

5.1 Importing fingerprint image dataset to Matlab workspace

%% import input dataset to create input cell array

pngfile= dir('.png'); % dir function creates fingerprint input dataset directory*

numfiles=length(pngfile);

mydata=cell(1,numfiles);

for k=1:numfiles

mydata{k}=imread(pngfile(k).name); % imread function reads the input dataset as a single cell from the directory *end*

inputarray=cell2mat(mydata); % cell2mat function convert input cell to matrix vector to return inputarray of [512 by 40650] for 80 fingerprint images

%% import input dataset to create target cell array

pngfile1=dir('.png'); % dir function creates fingerprint target dataset directory*

numfiles1=length(pngfile1);

mydata1=cell(1,numfiles1);

for k1=1:numfiles1

mydata1{k1}=imread (pngfile1(k1).name); % imread function reads the target dataset as a single cell from the directory

end

targetarray=cell2mat(mydata1); % cell2mat function convert cell to matrix vector to return targetarray of [512 by 40650] for 80 fingerprint images

5.2 Image processing Matlab implementation

The following is the Matlab image processing operations as shown in figure 3.2 for the input dataset, target dataset and fingerprint image to test the system.

5.2.1 Noise removal using Median Filter implementation

Medianfilteringinputarray= medfilt2(inputarray); % medfilt2 function preforms median operation on inputarray

Medianfilteringtargetarray=medfilt2(targetarray); % medfilt2 function preforms median operation on targetarray

%% Application of Median filtering on the test fingerprint image

% test fingerprint image to Matlab workspace using imread function, the imread return [512,512] vector array of unit8 values.

fingerprint1=imread('f0001_01.png');

noisyimage= imnoise(fingerprint1,'gaussian'); % imnoise function introduce noise into the test fingerprint image to be used to test network as a noisy input image

medfingerprint= medfilt2(noisyimage); % Median filtered fingerprint image

% to display noise image and median filtered image

figure

imshow (noisyimage)

figure

imshow (medfingerprint)

5.2.2 Normalization using contrast limited adaptive histogram equalization implementation

CLAHEinputarray=adapthisteq(Medianfilteringinputarray); % adapthisteq function to perform contrast limit adaptive histogram equalization on the median filtered inputarray

CLAHEtargetarray=adapthisteq(Medianfilteringtargetarray); % adapthisteq function to perform contrast limit adaptive histogram equalization on the median filtered targetarray

%% Application of CLAHE on the test fingerprint image

fingerprintCLAHE=adapthisteq(medfingerprint); % adapthisteq function to perform contrast limit adaptive histogram equalization on test fingerprint image

% to display test contrast limit adaptive histogram equalization enhanced image

figure

imshow (fingerprintCLAHE)

5.2.3 Binarazation and Thinning Process implementation

%% Binarization on the fingerprint images

BW=imbinarize(CLAHEinputarray,'adaptive'); % imbinarize function with adaptive method to performs binariation operation on the CLAHE filtered inputarray

BW1=imbinarize(CLAHEtargetarray,'adaptive'); % imbinarize function with adaptive method to performs binariation operation on the CLAHE filtered targetarray

%%Thinning operation using Morphological process

thin=bwmorph(BW,'thin'); % bwmorph function with 'thin' method to performs thinning operation on the binarized input dataset

thin2=bwmorph(BW1,'thin'); % bwmorph function with 'thin' method performs thinning operation on the binarized target dataset

% %Application of binarization and thinning process to the test fingerprint image

Binarizeimage=imbinarize(fingerprintCLAHE,'adaptive'); % imbinarize function with adaptive method to performs binariation operation on the test fingerprint image

% to display test binarized image

figure

imshow (Binarizeimage)

%%

Thinning=bwmorph(Binarizeimage,'thin'); % bwmorph function with 'thin' method performs thinning operation on the binarized test fingerprint image and returns logical array

```
% to display thinned image

figure

imshow (Thinning);

%% im2double functions returns a double precision array, thinning logical array is converted to double precision

thinarray=im2double(thin);

thin2array=im2double(thin2);

thinningarray=im2double(Thinning);
```

5.3 Gabor filtering and Local binary pattern feature extraction implementation

5.3.1 Orientations and frequencies for Gabor filter bank

```
imageSize=size(inputarray); % size function to determine the size of the input vector size, since the input and target size are equal no need to run for the target data set

numRows = imageSize(1);

numCols = imageSize(2);

Wavelengthmin= 4/sqrt(2);

Wavelengthmax=hypot(numRows,numCols);

n=floor(log2(Wavelengthmax/Wavelengthmin));

Wavelength=2.^(0:(n-2))*Wavelengthmin; % generating the frequency vector

Deltatheta=45;
```

Orientation=0:Deltatheta:(180- Deltatheta); % generating the orientation vector

%% creating Gabor filter bank to be convolved with the input images

wavelength=[6]; % selected from the frequency vector

orientation= [90^0]; % selected from the orientation vector

Gaborbank=gabor(wavelength,orientation); % gabor function creates the Gabor filter bank

5.3.2 Gabor filtering minutiae extraction implementation

Gabormag=imgaborfilt (thinarray,Gaborbank); % imgaborfilt function generates the extracted Gabor minutiae response from the thinned fingerprint input dataset

Gabormag1=imgaborfilt(thin2array,Gaborbank); % imgaborfilt function generates the extracted Gabor minutiae response from the thinned fingerprint target dataset

Gabormag2=imgaborfilt(thinningarray,Gaborbank); % imgaborfilt function generates the extracted Gabor minutiae response from fingerprint the test input dataset

Gabornoisy=imgaborfilt(noisyimage,Gaborbank);% feature extraction form noisyimage

5.3.3 Local Binary Pattern (LBP) feature extraction implementation

%% Feature extraction from Gabor filtering response using Local Binary Pattern (LBP) Feature to reduce the dimension

LBPinputarray=extractLBPFeatures (Gabormag); % extractLBPFeatures is the function for extracting the LBP features which returns a feature vector of single precision

LBPtargetarray=extractLBPFeatures (Gabormag1);

LBPtestarray=extractLBPFeatures (Gabormag2);

LBPnoisy=extractLBPFeatures(Gabornoisy);

%% im2double functions rescale the LBP feature vector array to a double precision to rearrange the array. Double precision array vector is used as input format to the network

inputarray2double=im2double(LBPinputarray);

testarray2double=im2double(LBPtestarray)

%%

targetbinary=imbinarize(LBPtargetarray);

targetarray2double=im2double(targetbinary);

5.4 Artificial neural network matching algorithm implementation

%% Neural network Application

nnstart % nnstart function is used to start the neural network GUI application for pattern recognition

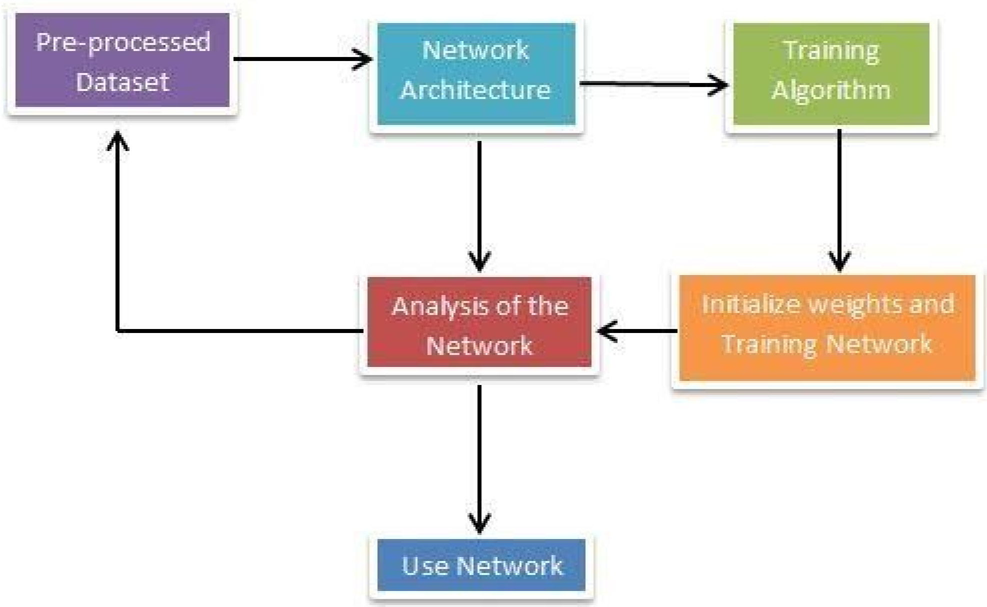

Figure: 5.2: Block diagram of the Neural Network implementation steps

The Matlab neural network toolbox has four tools as shown in figure 5.3 for different functions. The pattern recognition network used in the fingerprint recognition system for matching is a feed-forward multilayer network to be trained with Scaled Conjugate Gradient back propagation algorithm.

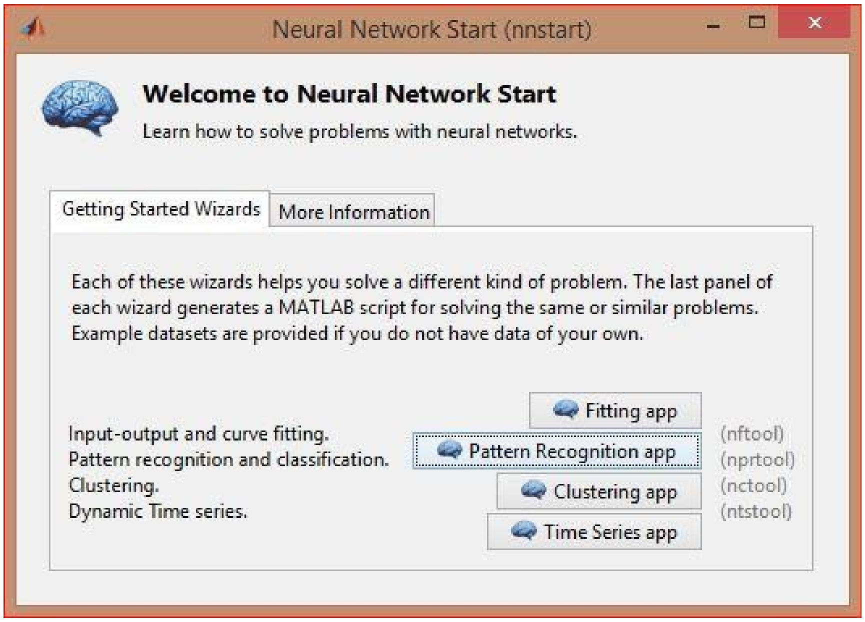

Figure 5.3: Matlab neural network application GUI

Step 1: Selection of the processed input and target dataset

Figure 5.4 gives the interface where the preprocessed input dataset (*inputarray2double*) vector and target dataset (*targetarray2double*) vector of the extracted minutiae as given in section 5.3.1 are selected for the network. The input and target dataset are numeric vectors of a 1 by 59 matrix.

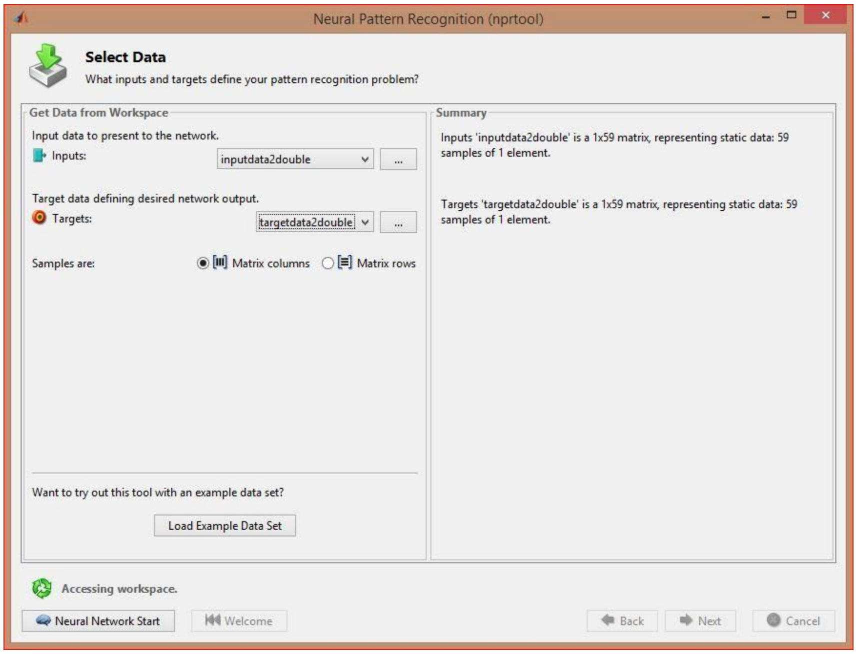

Figure 5.4: Data selection interface

Step 2: Configuration and initialization of the Network

When the datasets have been selected the next interface as shown in figure 5.5 appears. This interface is where the *inputarray2double* is randomly divided into training data, validation data and testing data using *dividerand* matlab function (Martins, et.al.,2016). The purpose of this division is just to evaluate the training performance, the network still learn the input dataset and the target dataset (Ali, Rosni ,Zong, 2011).

- Training data: this dataset is the largest portion it is 70% of the input minutiae. It automatically initializes the network weights and biases for training network.
- Validation data: the validation dataset is 15% of the input minutiae. It is used during the network learning process to evaluate the recognition ability of the network.
- Testing data: the test data is 15% of the input features. It is used by the network to provide independent test on how it has accurately learned the input datasets (Ali, Rosni ,Zong, 2011).

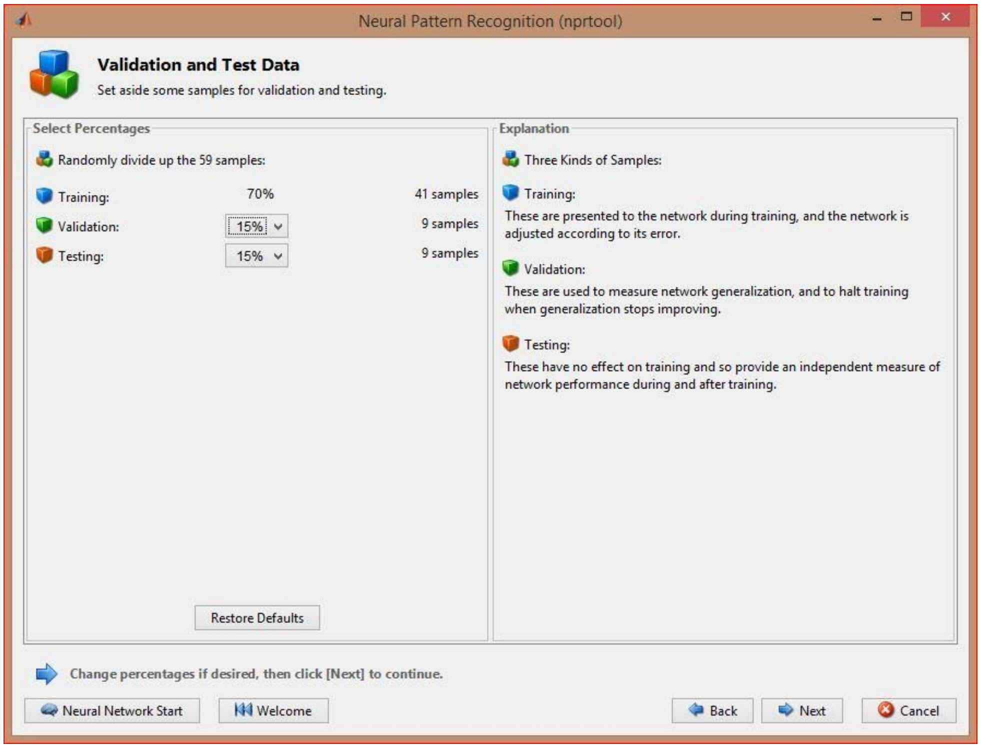

Figure 5.5: Network validation and test data interface

Step 3: Create the neural network

Figure 5.6 is where the network is created, the network architecture is a supervised learning multilayer feed-forward network with sigmoid activation function in the hidden layer and softmax (Matlab threshold activation function) in the output layer.

For the project implementation, the numbers of neurons are set to 20 and the network has been retrained 10 times to improve the performance of the network in recognizing the fingerprint images pattern. To ensure good performance, the network can be retrained several times and the numbers of neurons can be increased.

The network is very vulnerable to the numbers of neurons in the hidden layer; the very small number can induce underfitting that is under trained the network data while too many can induce overfitting that is over-trained the network data (Ali, Rosni, Zong, 2011).

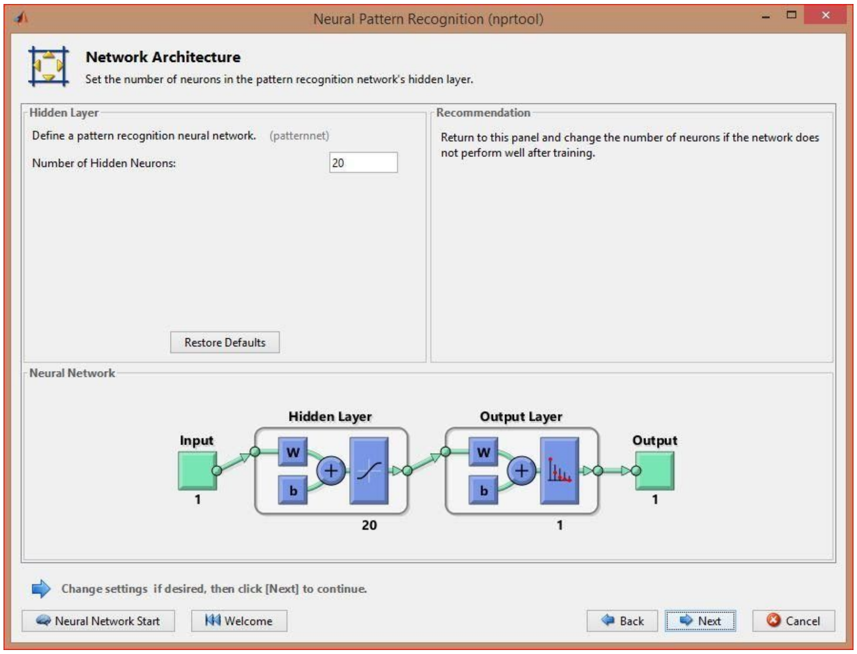

Figure 5.6: Network architecture interface

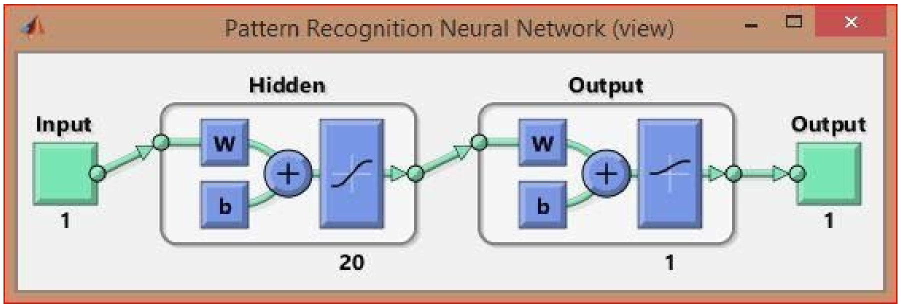

Figure 5.7: Trained network block diagram

Step 4: Training the network with the scaled conjugate gradient algorithm

Figure 5.8 gives the training interface where the Matlab training algorithm function *trainscg* is initialized to update the network weights and biases. The train button is clicked on to begin training of the network.

The training performances are measured with the following parameters as defined bellow,

- **Cross entropy (CE)**: measures the ability of the network to correctly predict its input. Very minimal values indicate good classification, while zero value means no error has occurred in the fingerprint pattern recognition. (Martins, et.al.,2016)
- **Percentage error (%e)**: show part of the training, validation and testing samples that are misclassified and it is expressed in percentage. A value of zero means there is no misclassification while a value of 100 indicates maximum misclassification. (Martins, et.al.,2016)

The following are the Matlab conditions required to stop training although it is done automatically by the network when one or more of the conditions are met. In appendix C table 1 presents summaries on each criterion;

- When it reaches the maximum number of Epoch
- When it exceeds the maximum time
- When the performance goal is reached
- When the performance gradient falls between the min_grad
- When the validation performance has increased more than maxi_fail. (Howard, et.al.,2002)

When training a network an optimal generalization performance with very minimal error is expected. However, the network is vulnerable to overfitting and underfitting as mentioned in step 3, to prevent these problems early stopping and regularization mechanism can be adopted.

- Early stopping is when training is stopped before the network automatically stops training. How to know when to use the early stopping method? It is when the network's 15% test dataset accuracy performance is poor which have indicated that overfitting or underfitting has occurred, figure 5.14 shows the test confusion matrix performance (Chi, Shie-Yui, n.d).
- The regularization mechanism is the process of setting the network training parameters specifications to reduce overfitting or underfitting and act as a regulator (Chi, Shie-Yui, n.d).

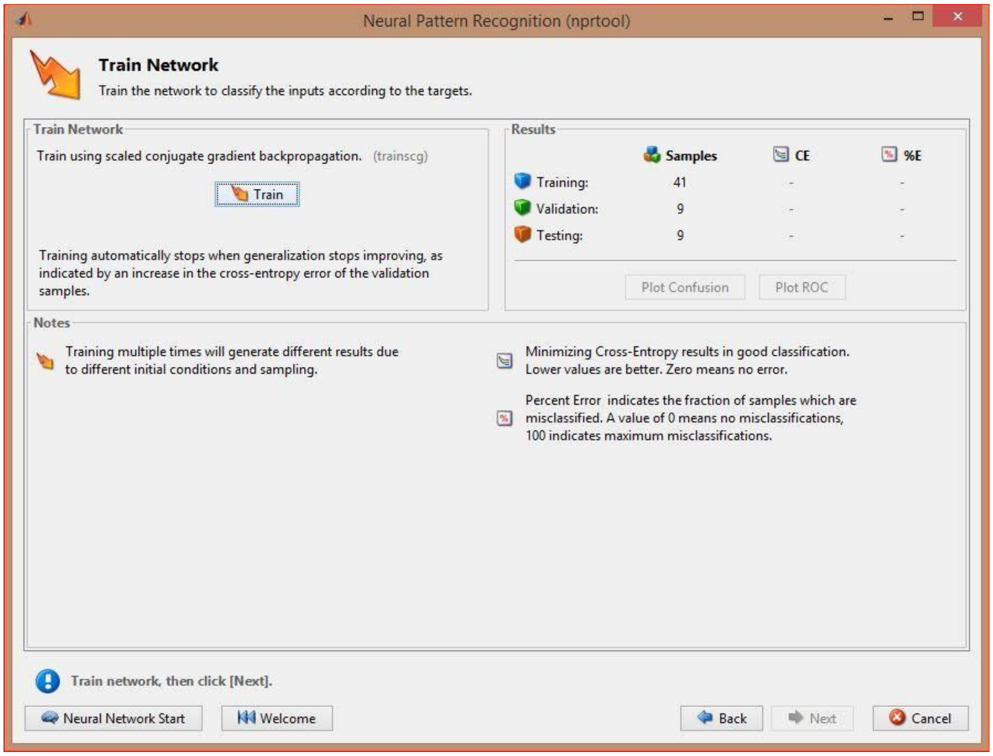

Figure 5.8: Training interface

On figure 5.9 interface the cross-entropy (CE) and per cent error (%e) outcome of the training are displayed; the result table 2 is presented in the appendix C. Also the network can be retrained using the retrain button on figure 5.9. In addition, for every training session that is when a retrain button is clicked the network starts with a different set of initial weights and biases this returns a different network performance.

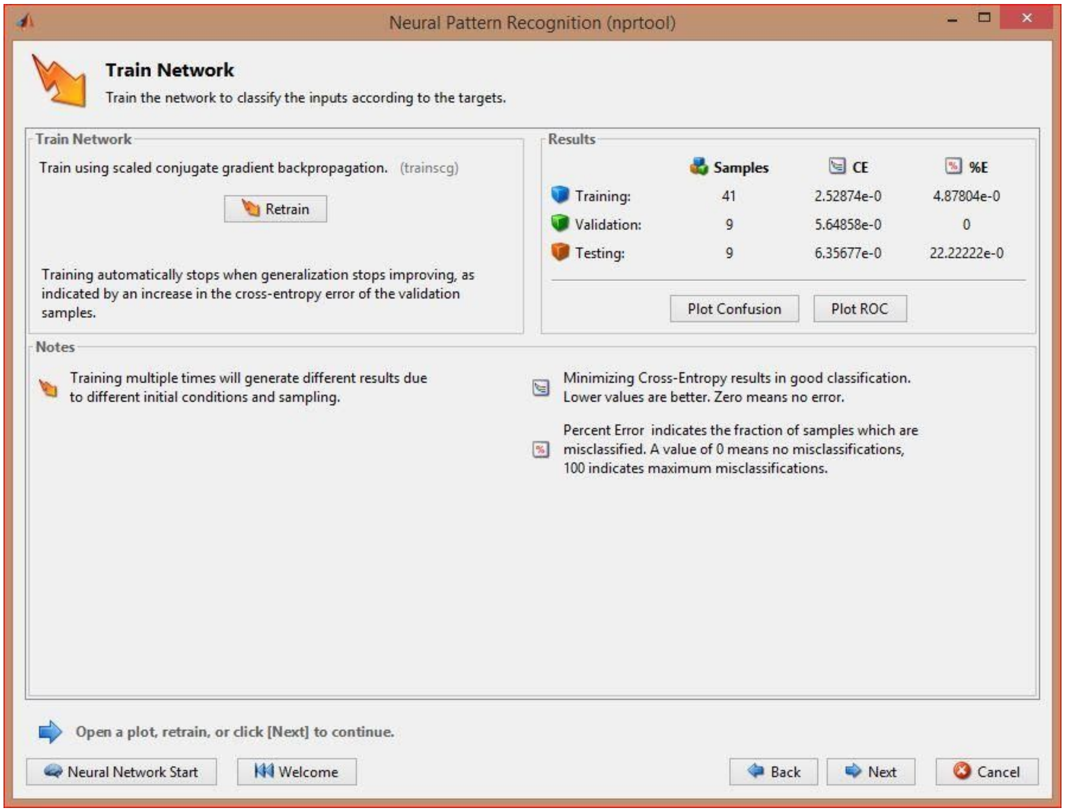

Figure 5.9: Re-training interface

Step 5: Training performance index evaluation

When the train button from figure 5.8 is activated, figure 5.10 interface immediately pops up. This window is known as the performance index window, which is used to evaluate the performance of the network for the fingerprints recognition system. The training was terminated when the validation error increased for six iterations which represent the Matlab default validation checks for the training. During training, the progress is continuously updated. The number of validation check represents the number of successive iterations that the validation performance fails to decrease which is at 65 iterations for this implementation, see figure 6.10 (Martins, et.al.,2016)

The performance index window is divided into four sections, the Network block diagram, the algorithms, the training parameters and the latter includes Performance plot, Training state plot, Error histogram plot, Confusion matrix plot and Receiver operating characteristics for visual assessment of the network performance.

The default performance function for feed-forward networks in Matlab is measured in mean squared error (MSE) that is the average squared error between network output and the target output. MSE is defined as;

$F = mse = \frac{1}{N} \sum_{i=1}^{N} (ti - ai)^2$ *where* t_i is the target output and a_i is the network output (Martins, et.al., 1995)

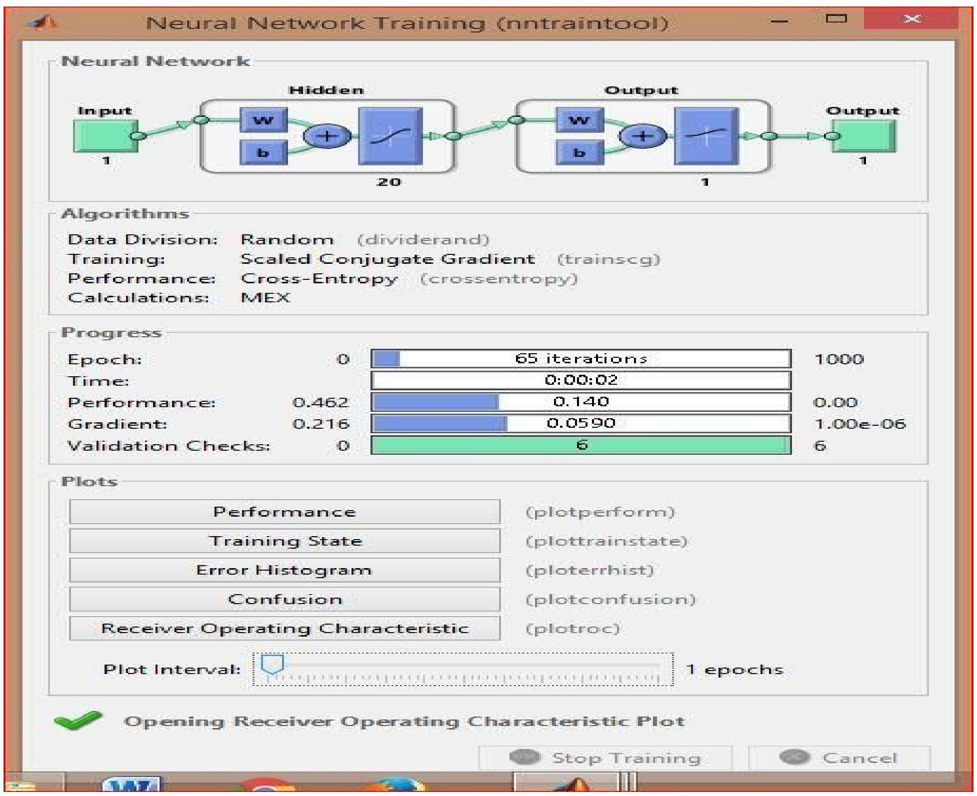

Figure 5.10: Performance index interface

Figure 5.11 shows the performance plot, it indicates with legends the plot of training errors in blue, validation error in green, and testing error in red, while the best fit in broken lines. The best validation performance is 0.0051183 at 59 epochs the intersection is indicated with the green circle. Since the final mean square error 0.0051183 is small and the training and test curves have almost similar characteristics with no overfitting occurring, this indicates that the network performance is moderate (Martins, et.al., 1995). The best fit epoch of 59 is when the validation performance reaches the minimum.

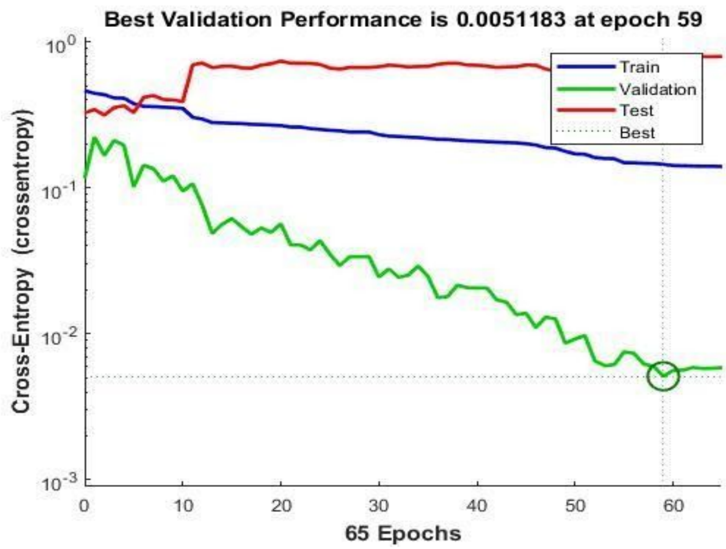

Figure 5.11: Network performance plot

In figure 5.11, the progress of other training variables such as the gradient magnitude which is 0.059035at iterations 65 and the number of validation checks which is 6 are shown.

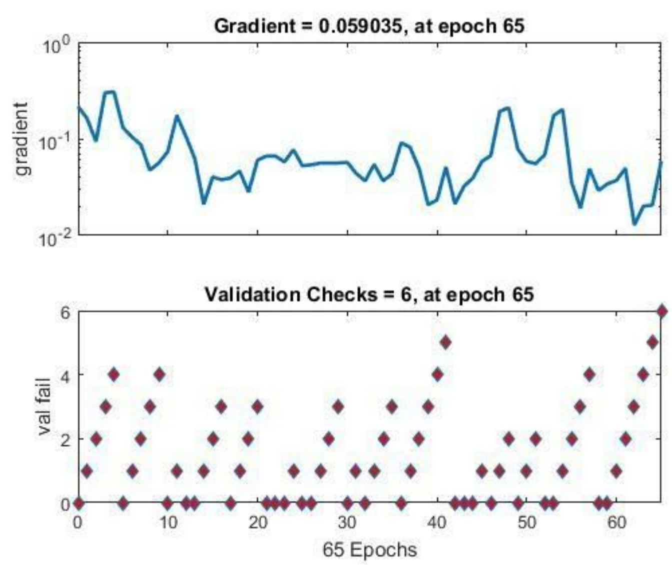

Figure 5.12: Training state plot

Figure 5.13 is the error histogram that displays the pattern recognition error distribution. The histogram shows the measure of error between the network output and target input (*Errors=Targets data-output data)*. The legend colours represent the data division as explained in step 2, the errors falls between -0.9382 and 0.8641 on the graphs. When the training instance reaches 38, the minimum error is measured to be 0.0104 that is where the error line in orange colour falls on the graph. However, the histogram is used to check the accuracy of the network performance.

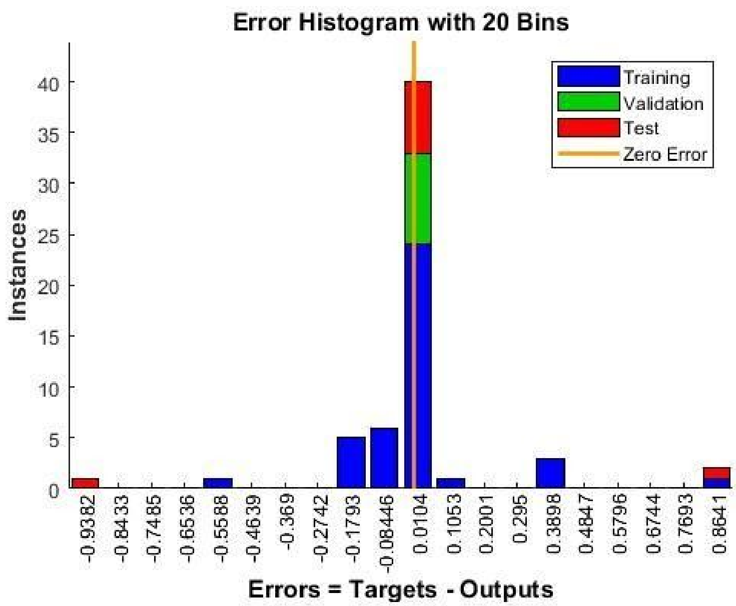

Figure 5.13: Error histogram plot

The confusion matrix also known as the classification table shows the performance matrix for the training, testing and validation data along with the overall recognition performance. The table columns represent the target data class and the rows represent the network output class which classifies the network performance to it input. The network then classified the target input features into an output of classes 0 and 1 to classify its responses to a given input. The confusion matrix evaluates the network responses to it input.

The training confusion matrix shows the percentages of the training data in the green diagonal boxes, 82.9% for target class 0 and 12.2% for target class 1 that are correctly recognized while the incorrect data is represented in the red diagonal boxes, 2.4% for target class 0 and 2.4% for target class 1. Same is applicable to the validation confusion matrix, the testing confusion matrix and all confusion matrix.

The latter shows that the network performance is accurate due to high numbers of correct responses in the green diagonal squares and low numbers of incorrect responses in red diagonal squares. The lower right blue square illustrate the overall accuracy of the network, the green value 93.2% shows the overall training percentage of the cases that are correctly classified and the red value at 6.8% shows the overall training percentage of cases that are misclassified that is are rejected by the network. This outcome from the network indicates accurate learning of the fingerprint images.

Figure 5.14: Confusion matrix plot

The Receiver operating characteristics (ROC) curve as shown in figure 5.15 is a graphical curve that illustrates the comparison of two operating characteristics FAR and FRR in the case of a biometric recognition system as its set threshold is varied. It is the plotting of true positive rate (sensitivity) versus the false-positive rate (specificity) (Martins, et.al.,2016).

The colour blue in each axis represents the ROC curve, the true positive rate of the network recognizes the input data and the false positive rate is set as a threshold which varies. The zigzag points at upper left corner indicate a reasonable performance, and that the network has learned almost to 100% (represents a straight line at the edge) to recognize fingerprint pattern. From this plot the network output is compared against the threshold value that ranges from-1 to 1. (0, 1) is the ideal points for ROC curve represented by the diagonal line, 0 correspond to no false-positive value whereas as 1 correspond to a true positive value (Martins, et.al., 1995).

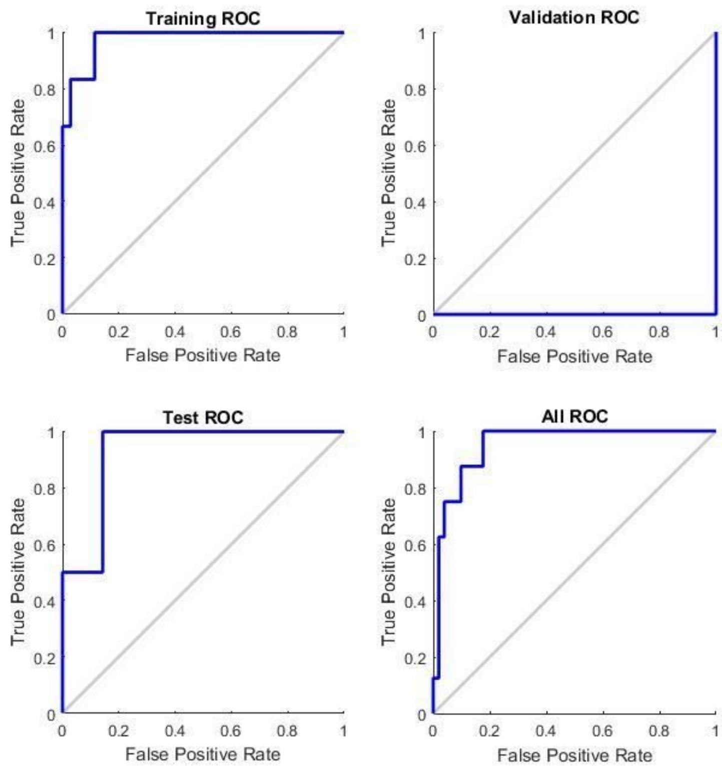

Figure 5.15: Receiver operating characteristics (ROC) curve plot

5.4.1 Testing neural network matching algorithm

The testing network uses the confusion matrix to evaluate the network performance to a new set of input. Testing is a post-training analysis to determine the success of network training. Two fingerprints input are used to test the network accuracy for the fingerprint matching.

Input 1: known user *testarray2double (from section 5.3.1)*

Input 2: known noisy image user *noisy image (from section 5.2)*

The false acceptance rate (FAR) and false rejection rate (FRR) is used to evaluate the network for the fingerprint recognition system. False acceptance rate percentage is when the system incorrectly accepts the unknown user as authentic. It is the ratio of acceptance number that is the correctly classified responses to the number of identification attempt. This is also referred to as a type II error in an artificial neural network.

False rejection rate is when the system incorrectly rejects access from a known user. It is the ratio of the number of false rejection of the number of identification attempt. This is referred to as type I error in an artificial neural network. The identification is examined as the similarity between the trained minutiae and the input data.

Step 1: Selecting test data

Figure 7.16 shows the test interface, where the network is tested with a test input data which is selected from Matlab workspace. If the performance is not satisfactory the network can be retrained. If training performance is good and the test performance is worst it could indicate that overfitting has occurred while training.

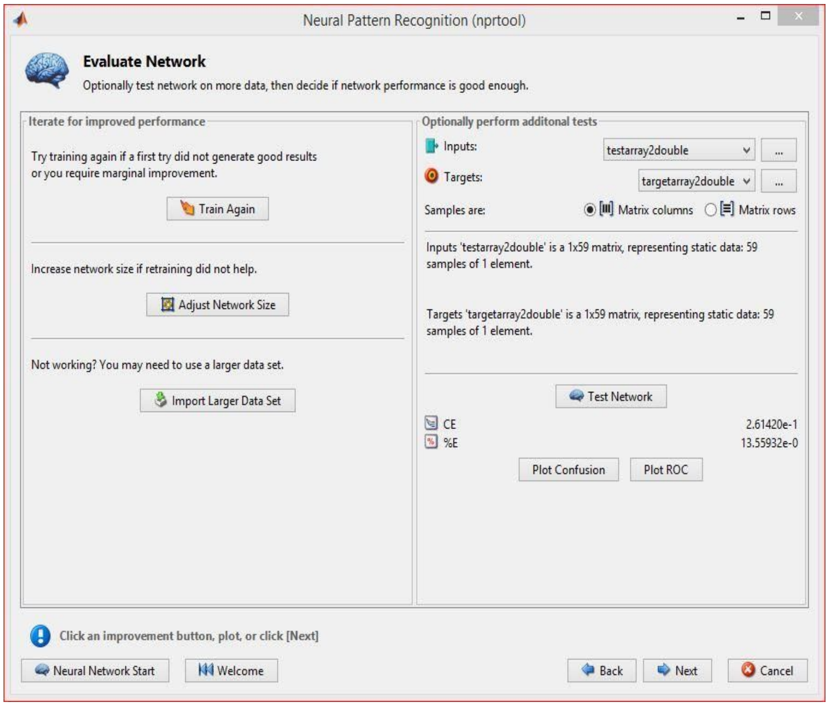

Figure 5.16: Test network interface

Step 2: Generating the experimental result from the test network interface

Figure 5.17 display the confusion matrix for the known user. The target class and output class of the input minutiae are classified into two that is 0 and 1. 47 (79.7%) of the samples belong to class one represented as 0 in the green diagonal square, 6 (10.2%) of the sample belonging to class 2 represented as 1 and are correctly classified minutiae.

The red diagonal cells are the misclassified samples for each class. Where the lower left shows that 4 (6.8%) of the samples from class 1 were misclassified by the network as class 2. If class one (0) is the positive outcome of the recognition system then 6.8% is the false acceptance rate that is the false negatives (type II errors).

The upper right cell shows that 2 (3.4%) input sample from class 2 was misclassified by the network as class 1, this indicates the false rejection rate of the recognition system that is the false positives (type I errors). The overall accuracy of the test network shown on the bottom right blue square gives a total of 89.8% correct response which shows that the fingerprint is accepted and 10.2% incorrect response.

If 3.4% is higher than 10.2% then the system has falsely rejected the fingerprint, while if 6.8% is higher than 79.7% then it has falsely accepted a fingerprint. The recognition performance of the fingerprint images recognition system is acceptable in this case.

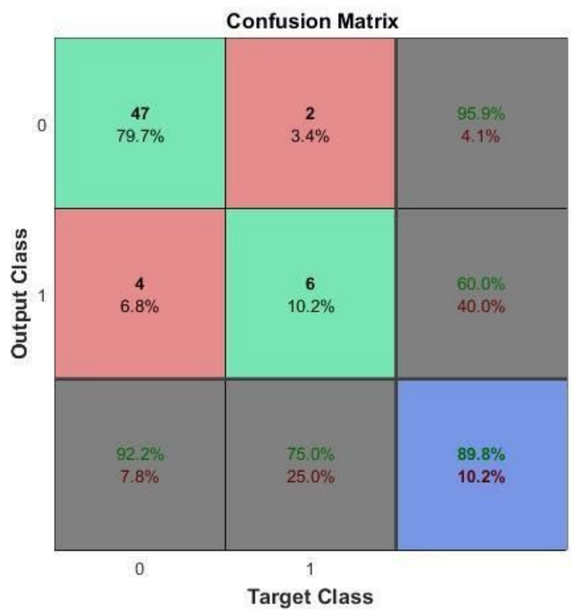

Figure 5.17: Test confusion matrix plot for known user fingerprint

Figure 5.18 shows the confusion matrix for known user noisy fingerprint image, the recognition rate is at 74.6% which shows that the network still recognized the fingerprint image. Although, the network may not recognise the image, if the image is blurry, of low contrast and even noisy image. As the network performance drops with the known noisy fingerprint compared to the processed known user image. Table 4 in appendix C shows the neural network matching scores for the test input data.

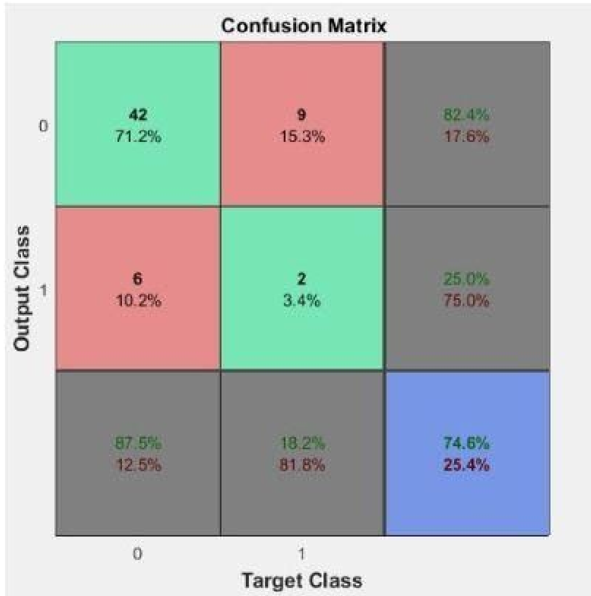

Figure 5.18: Test confusion matrix plot for noisy known fingerprint

Chapter 6

Conclusion and Recommendation

The project research focused on the implementation of fingerprint recognition system using a neural network algorithm has addressed. One major contribution is to achieve reliable minutiae matching of fingerprints in a recognition system. A Multilayer neural network is trained with Scaled Conjugate Gradient Descent algorithm. Other methodology proposed includes image processing, feature extraction of the fingerprint images.

The neural network performance index was analyzed using the confusion matrix, error histogram, and the receiving operation curve. From the network training result, the network indicates 6.8% is the false acceptance rate and 3.4% false rejection rate with 89.8% correct response. This shows that the neural network approach can provide a reliable, fast convergence and good performance when implemented in a fingerprint recognition system.

The neural network backpropagation algorithms were analyzed while the Gabor filtering and local binary pattern feature extraction algorithms and image processing algorithms such as Median filtering, CLAHE and Binarisation and Thinning were also discussed and implemented in Matlab.

Furthermore, it is recommended that for more accurate performance of the network the training database should be increased. 2D Gabor filtering can be used for dual purpose, it can be directly applied on raw fingerprint images to enhance and extract minutiae. The design of Convolutional Neural Network (CNN) can be implemented for the fingerprint recognition system. It is more efficient for noisy latent (incomplete) images, low-quality images and capable of differentiating between spurious and real fingerprint image.

References

1. Simon, H., (2005). Neural Networks A Comprehensive Foundation. India: Pearson Prentice Hall.
2. Martin,T.H.,Howard,B.D.,Mark,H.B., (1995).Neural Network Design.pp.734.DOI: 10.1007/1-84628-303-5.
3. Pietikainen,M.,Hadid,A.,Zhao,G.,Ahonen,T.,(2011).Computer Vision using Local Binary Pattern.pp13-14.DOI: 10.1007/978-0-85729-748-8
4. Howard,D.,Mark,B.,(2002). Neural Network Toolbox User's Guide. Retrieved from http://www.mathworks.com/help/nnet
5. Martin,T.H.,Howard,B.D.,Mark,H.B., (2016) Neural Network Toolbox™ Getting Started Guide. Retrieved from http://www.mathworks.com/help/nnet
6. Christopher, M.B., (1995). Neural Networks for Pattern Recognition. Oxford: Clarendon Press.
7. Ivan,N.S.,Danilo,H.S.,Rogerio,A.F.,Luisa,H.B.,Silas,F.R.,(2017).Artificial Neural Network Architecture and Training Process. pp 21-28.DOI: 10.1007/978-3-319-43162-8
8. Greenberg, S., Aladjem, M., & Kogan, D. (2005). Fingerprint Image Enhancement using Filtering Techniques. Retrieved from https://doi.org/10.1109/SIU.2005.1567662
9. Gonzalez R.C.,Richard E. W.,Steven L.E.,(2016) Digital Image Processing Using Matlab. Retrieved from https://docs.google.com/file/d/0B_9nttbyYh5MNmE3Y2YxNzgtODk2ZC00ZThjLTg2ODctNDFkNjJjMWExNjE1/
10. Gonzalez R.C.,Richard E. W.,(2008) Digital Image processing. Retrieved from http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_2ndEd.pdf
11. Maio, D., Jain, A. K., Prabhakar, S., Maltoni, D., & Sacchi, V. (2009). Handbook of Fingerprint Recognition.https://link.springer.com/book/10.1007%2F978-1-84882-254-2
12. Bovik, A., (2009). The Essential Guide to Image Processing. London.Uk: Academic press publication.

13. Shaheed,M.,.(2004). Performance analysis of 4 types of conjugate gradient algorithms in the nonlinear dynamic modeling of a TRMS using feed-forward neural: IEEE International Conference on Systems, Man and Cybernetics.(pp 5985-5990). Retrieved from http://ieeexplore.ieee.org/document/01401153/
14. Mohamed, E., Christophe, C., (2012). Evaluation of Biometrics System. https://hal.archives-ouvertes.fr/hal-00990617/file/InTech-Evaluation_of_biometric_systems.pdf
15. Parker,J.R.,(2011).Algorithms for Image Processing and Computer Vision. USA: John Wiley&Sons
16. Puneet.G.B.,Naresh.K.G.,(2013) Binarization Techniques used for Gray scale Images. Retrieved from https://pdfs.semanticscholar.org/c58f/8ab48711f0d5596be78c7f403027030ecc5b.pdf

17. Da Silva,I.N., Hernane,S., Andrade,F.R., Liboni,L.H., dos ReisAlves,S.F.,(2017).Artificial Neural Networks.pp21-28.DOI 10.1007/978-3-319-43162-8_2
18. Di,H., Caifeng,S., Mohsen,A., Yunhong,W.,Liming,C.,.(nd) Local Binary Patterns and Its Application to Facial Image Analysis: A Survey. Retrieved from http://ieeexplore.ieee.org/document/5739539/
19. Navrit,K.J.,Amit,K.,(2011). A Novel Method for Fingerprint Core Point Detection. International Journal of Scientific & Engineering Research Volume 2, Issue 4.Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.301.481
20. Ruxin,W., Congying,H., Yanping.W.,Tiande.G.,(2014) Fingerprint Classification Based on Depth Neural Network. Retrieved from https://www.researchgate.net/publication/265852659_Fingerprint_Classification_Based_on_Depth_Neural_Network
21. Romulo.F.L.,Jessyca,A.B.,Jermana,L.M.,Edson,C.N.,Auzuir,R.A.,(2014). Techniques of Binarization, Thinning and Feature Extraction Applied to a Fingerprint System. International Journal of Computer Applications (0975 8887) Volume 103.Retreived from http://research.ijcaonline.org/volume103/number10/pxc3899291.pdf

22. Ilonene,J.,Kamarainen,J.K.,Kalviainen,H.,(2005). Efficient Computation of Gabor Features. Retrieved from http://www2.it.lut.fi/project/simplegabor/downloads/laitosrap100.pdf
23. Alaa,A.A.,Ghazali,S.,(2014). Fingerprint Classification Techniques: A Review. IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 1.Retrieved from https://www.ijcsi.org/papers/IJCSI-11-1-1-111-122.pdf
24. Moller,M.,(1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Retrieved from http://www.sciencedirect.com/science/article/pii/S0893608005800565
25. The History of Fingerprint (2017). Retrieved from http://www.onin.com/fp/fphistory.html
26. Tinku,A.,Ajoy,K.R.,(2005).Image Processing Principles and Applications.: United State of America: USA: John Wiley&Sons
27. Biometrics (2007). In Wikipedia. Retrieved May 2017from https://en.wikipedia.org/wiki/Biometrics
28. Zhi-Qiang,L.,Jin-Hai,C.,Richard,B.,(2003).Handwritten Recognition: Soft Computing and Probabilistic Approaches. New york: NY: Springer
29. Naim, N.,Yassin, A.,Zamri, W., Ameerul,W.,&Sarnin, S.(2011).MySQL Database for Storage of Fingerprint Data:13th International Conference on Modeling and Simulation.(pp293-298). Retrieved from http://ieeexplore.ieee.org/document/5754229/
30. Ryu, C., Kong, S. G., & Kim, H. (2011). Enhancement of feature extraction for low-quality fingerprint images using stochastic resonance. Pattern Recognition Letters, 32, 107–113. https://doi.org/10.1016/j.patrec.2010.09.008
31. M. Sepasian, W. Balachandran and C. Mares.(2008).Image Enhancement for Fingerprint Minutiae-Based Algorithms Using CLAHE, Standard Deviation Analysis and Sliding Neighborhood. Retrieved from https://www.researchgate.net/publication/44262481
32. Yin, Y., Wang, Y., & Yang, X. (2005).Fingerprint Image Segmentation Based on Quadric Surface Model.Retrieved from http://link.springer.com/chapter/10.1007%2F11527923_67
33. Image Filtering (2010).Retrieved from https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering_2up.pdf

34. National Institute of Science and Technology (NIST)(2017). Biometric Special Databases and Software. [database record] Retrieved from https://www.nist.gov/itl/iad/image-group/resources/biometric-special-databases-and-software
35. Image segmentation.(n.d). In Wikipedia. Retrieved May 2017 from https://en.wikipedia.org/wiki/Image_segmentation
36. Normalization.(n.d). In Wikipedia. Retrieved May 2017 from https://en.wikipedia.org/wiki/Normalization_(image_processing)
37. Ali K.,Rosni A.,Zong W.G.,(2011).Artificial Neural Network Training and Software Implementation Techniques. Retrieved from https://ebookcentral.proquest.com/lib/portsmouth-ebooks/reader.action?docID=3021387
38. Laurene V. F.,(1994).Fundamentals Of Neural Networks: Architecture, Algorithm and Application. Retrieved from http://www.csbdu.in/csbdu-old/pdf/Fundamentals%20Of%20Neural%20Networks.pdf
39. Chi D.D, Shie-Yui L.,(n.d) Generalization for multilayer neural network bayesian regularization or early. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.4827&rep=rep1&type=pdf
40. Artificial intelligent neural networks.(n.d). Retrieved from https://www.tutorialspoint.com/artificial_intelligence/pdf/artificial_intelligence_neural_networks.pdf
41. Local Binary pattern.(n.d). In Wikipedia. Retrieved May 2017 from https://en.wikipedia.org/wiki/Local_binary_patterns

Appendix A: Achievements

A.1: Project Achievement

1. I have been able to establish a fast computational and reliable matching algorithm, yielding 93.8% accuracy in classification for fingerprint recognition system using Scaled Conjugate Gradient neural network algorithm from Matlab neural network toolbox.

2. I have been able to effectively use the Matlab Image processing toolbox to implement Median filtering, Contrast limit adaptive histogram equalization, Binarization and Thinning algorithms on the fingerprint images before feature extraction.

3. I have been able to use the computer vision toolbox to implement Local binary pattern and also the Gabor filtering algorithms for fingerprint feature extraction. The extracted features serve as input into the Neural Network.

A.2: Personal Achievement

1. I learned to set up the following Matlab toolbox applications;
- Image processing toolbox
- Computer vision toolbox
- Neural network toolbox
2. I learned to manage time and resources efficiently
3. I learned to be productive and organized
4. I acquired problem solving skill, time management skill and critical thinking

Appendix B: MATLAB Codes

B.1 Matlab syntax to import image data to Matlab work space

```
numfiles=length(pngfile);

mydata=cell(1,numfiles);

for k=1:numfiles

mydata{k}=imread(pngfile(k).name);

end

inputarray=cell2mat(mydata);
```

B.2 Image processing codes

Matlab syntax function for Medianfiltering

B= medfilt2 (A)

Where A is the input image, medfilt2 is median filtering function, B is the median filtering output. (Gonzalez,et,al,.2016). Figure 3.5 is an example of a median filtered fingerprint image

Matlab synatax function for Contrast limit adaptive histogram equalization

clahe = adpthiseq(I)

Where clahe is the CLAHE image output, I is the input image, adpthisteq is the CLAHE function (Gonzalez,et,al,.2016).

Matlab syntax function for Binariazation process

BW= imbinarize (I, 'adaptive')

Where I is the input image, adaptive is the method, imbinarize selects the threshold value to reduce the black and white infraclass variance of the image intensity (Gonzalez,et,al,.2016).

Matlab syntax function for Thinning process

Thinnning= bwmorph(BW,'thin').

Thin is method, bwmorph is thinning function, Bw is the binary image, Thinning is the output response (Gonzalez,et,al,.2016)

B.3 Feature extraction codes

Matlab syntax function for Gabor filter bank and Gabor filtering feature extraction

Gabor bank= gabor (frequency, orientation)

Gabor magnitude = imgaborfilt (A, gabor bank);

Imgaborfilt is the Gabor function, A is the input gray image, Gabor magnitude is the feature response.

Matlab syntax function for Local binary pattern feature

LBP features= extractLBPfeatures(I),

Where I is the input image, LBP feature is the uniform LBP features of 1by N vector, N represents number of features vector length.

B.3 Neural network matlab codes

Matlab syntax function to start neural network toolbox

nnstart

The script created from the interface shown below contains the Matlab command line functionality. It reproduces the training steps of the neural network pattern recognition.

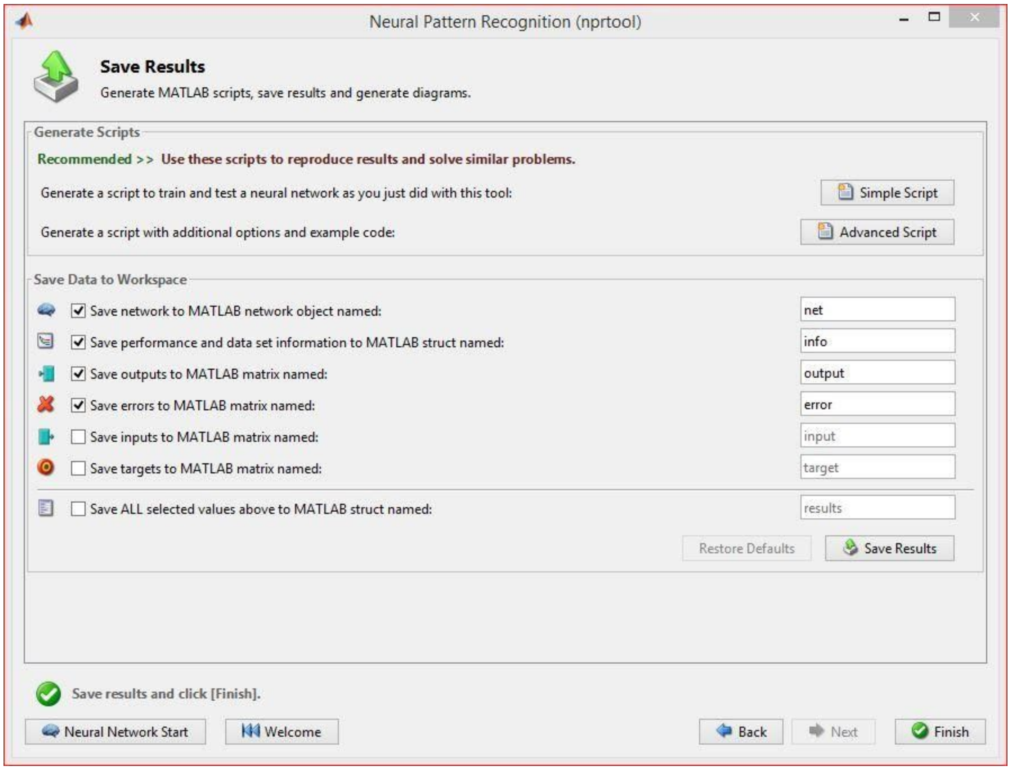

Figure 5.19: Matlab script interface

```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by Neural Pattern Recognition app
% Created 06-Apr-2017 20:07:23
% This script assumes these variables are defined:
%   inputarray2double - input data.
%   targetarray2double - target data.

x = inputarray2double;
t = targetarray2double;
```

```
% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory, suitable in low memory situations.

trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.

% Create a Pattern Recognition Network

hiddenLayerSize = 20;
net = patternnet(hiddenLayerSize, trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess

net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide

net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
```

```
net.performFcn = 'crossentropy';  % Cross-Entropy

% Choose Plot Functions
% For a list of all plot functions type: help nnplot

net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotconfusion', 'plotroc'};

% Train the Network

[net,tr] = train(net,x,t);

% Test the Network

y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);

% Recalculate Training, Validation and Test Performance

trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
```

```
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotconfusion(t,y)
%figure, plotroc(t,y)

% Deployment
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
   % Generate MATLAB function for neural network for application
   % deployment in MATLAB scripts or with MATLAB Compiler and Builder
   % tools, or simply to examine the calculations your trained neural
   % network performs.
   genFunction(net,'myNeuralNetworkFunction');
   y = myNeuralNetworkFunction(x);
end
if (false)

   % Generate a matrix-only MATLAB function for neural network code
   % generation with MATLAB Coder tools.

genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
   y = myNeuralNetworkFunction(x);
end
if (false)
```

```
    % Generate a Simulink diagram for simulation or deployment with.
    % Simulink Coder tools.

gensim(net);
end
```

Appendix C Neural Network Pattern Recognition Results

Training algorithm parameter	Descriptions
Min_grad	Minimum gradient
Max_fail	Maximum number of validation increases
Time	Maximum training time
Goal	Minimum performance value
Epochs	Maximum number of training epochs (iterations)
Lr	Learning rate
Max_perf_inc	Maximum performance increase

Table 1: Training Algorithm parameters (Howard, et.al., 2002)

	Samples	Cross entropy	Percentage Error (%)
Training	41	2.52874e-0	4.87804e-0
Validation	9	5.64858e-0	0
Testing	9	6.35677e-0	22.22222e-0

Table 2: Trained network cross entropy and percentage error results

Parameters	Neural network
Input layer	1
Hidden layer	1
Output layer	1
Minimum gradient decent	0.0051183
Number of neurons	20
Number of epoch	59
Minimum error	0.0104

Table 3: Neural network parameters

Testing	Neural network matching score
Known user fingerprint	89.8%
Known user noisy fingerprint	74.6%

Table 4: Neural network test matching score

Appendix D Monitoring Form

Monitoring Report Form (DL)

ENG601d2 Individual Project

PROJECT MONITORING FORM

Student: Kafayat Adunola Adeoye	Student Number:UP796445
Supervisor: Dr Abdsamad Benkrid	
Title: Fingerprint Recognition System using Neural Network	
Award: BEng Electronic System Eng'g (DL)	Year: 2017

Note Students should consult their supervisors and keep them informed about the progress of their work.

Date	Deliverable	Comments	Supervisors Initials
23rd Jan 2017	Project Proposals and Allocation including Project Plan & Ethics	Project proposal form was approved and signed by the supervisor. The ethic review	**AB**

	Review Cert. to be completed	form and project proposal form were submitted via Moodle	
23rd Feb 2017	Progress Review with supervisor	Project log document with timelines was shared with the supervisor	AB
24th Feb 2017	**Progress/Preliminary/Interim Report to be submitted** (Optional)		
17th March 2017	Progress Review with supervisor	I sent to my supervisor project background chapter and Implementation results and my supervisor replied with feedback	AB
3rd – 21st April	Easter Recess		
23rd April 2017	Progress Review with supervisor	I sent to my supervisor project report chapters and project video sample and my supervisor replied with feedbacks	AB
22nd May 2017	**Full Report Submission date**	**Project submitted**	AB