

INTELLIGENT FINGERPRINT RECOGNITION SYSTEM

SY MOHD SYATHIR BIN SY ALI ZAINOL ABIDIN

UNIVERSITI MALAYSIA PAHANG

SY MOHD SYATHIR BIN SY ALI ZAINOL ABIDIN

BACHELOR OF ELECTRICAL ENGINEERING (POWER SYSTEMS)

2007

UMP

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: INTELLIGENT FINGERPRINT RECOGNITION SYSTEM

SESI PENGAJIAN: 2007/2008

Saya SY MOHD SYATHIR BIN SY ALI ZAINOL ABIDIN (850423-02-5105)
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/Sarjana /Doktor Falsafah)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

4-D TAMAN BAHAGIA
06000 JITRA
KEDAH

NOR MANIHA ABDUL GHANI
(Nama Penyelia)

Tarikh: 28 NOVEMBER 2007

Tarikh: : 28 NOVEMBER 2007

- CATATAN:
- * Potong yang tidak berkenaan.
 - ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
 - ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

“I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)”

Signature : _____

Name : NOR MANIHA BT ABDUL GHANI

Date : 28 NOVEMBER 2007

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : _____

Author : SY MOHD SYATHIR BIN SY ALI ZAINOL
ABIDIN

Date : 28 NOVEMBER 2007

DEDICATION

“To my beloved parents,

Mr Syed Ali Zainol Abidin and Mrs Sharifah Seha

And those who love me”

ACKNOWLEDGEMENT

Alhamdulillah, His Willingness has made it possible for me to complete the final year project in time. I would like to take this opportunity to express gratitude to my dedicated supervisor, Mrs Nor Maniha binti Abdul Ghani for guiding me this project at every stage with clarity and that priceless gift of getting things done by sharing his valuable ideas as well as share his knowledge. I would also like to thank to all UMP lecturers and electrical technicians whom had helped directly or indirectly in what so ever manner thus making this project a reality.

Not forgotten to my heartfelt thanks to my beloved parents, Mr Syed Ali Zainol Abidin and Mrs Sharifah Seha and also the rest of my dearest family whom always support and prays on me throughout this project. Also to my best colleagues for their openhandedly and kindly guided, assisted, and supported and encouraged me to make this project successful.

Their blessing gave me the high-spirit and strength to face any problem occurred and to overcome them rightly. The great cooperation, kindheartedness and readiness to share worth experiences that have been shown by them will be always appreciated and treasured by me, thank you.

ABSTRACT

The purpose of this project is to design and develop a pattern recognition system with using Artificial Neural Network (ANN) that can recognize the type of image based on the features extracted from the choose image. This system which can fully recognizing the types of the data had been add in the data storage or called as training data. The Graphic User Interface in Neural Network toolbox is used. This is the alternative way to change the common usage of the MATLAB which are use the command insert at command window. From this kind of system, we just need to insert the features data or training data. The recognition done after we insert the test data. The system will recognize whether the output is match with the training data. Then output will produce a kind of graph that describes the feature of the data which is same as the training data.

ABSTRAK

Tujuan projek ini dilakukan adalah untuk mereka dan menghasilkan sistem pengesanan corak dengan menggunakan Artificial Neural Network (ANN) dimana ia akan mengesan jenis gambar berpandukan ciri-ciri akstrak dari gambar yang dipilih. Sistem ini sepenuhnya boleh mengesan jenis-jenis data yang telah disimpan dalam memori atau dipanggil data latihan. Graphic User Interface dalam Neural Network toolbox digunakan. Ini adalah cara alternatif untuk menggantikan cara biasa dalam MATLAB iaitu hanya meletakkan arahan ke jendela arahan (command window). Untuk sistem jenis ini, kita hanya meletakkan cirian data atau data latihan. Sistem ini akan mengesan dengan hasil keluaran sama dengan data ujian. Hasil keluaran graph menerangkan cirian data adalah sama dengan data latihan.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	i
	DEDICATION	iv
	ACKNOWLEDGEMENT	v
	ABSTRACT	vi
	ABSTRAK	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLE	xi
	LIST OF FIGURES	xii
	LIST OF SYMBOLS	xiv
	LIST OF ABBREVIATIONS	xv
	LIST OF APPENDICES	xvi
1	INTRODUCTION	1
	1.1 Fingerprint	1
	1.2 Artificial Intelligent (AI)	2
	1.3 Neural Network	2
	1.4 Graphical User Interface (GUI)	4
	1.5 Problem Statement	4
	1.6 Objectives	5
	1.7 Scope of Project	5
	1.8 Literature review	6
	1.8.1 Fingerprint	6
	1.8.2 Artificial Intelligent (AI)	7

1.8.3	Neural Network Toolbox	9
1.8.4	Pattern Recognition	15
1.8.5	Graphical User Interface (GUI) in GUIDE	16
1.9	Methodology	17
1.9.1	Introduction	17
1.9.2	Methodology	18
1.9.3	Software Design	18
1.9.4	Software Procedure	19
1.9.5	Result and Analysis	20
1.9.6	Conclusion	20
2	SYSTEM MODEL	21
2.1	Introduction	21
2.2	System Model	21
3	SOFTWARE	24
3.1	Introduction	24
3.2	Training Data Table	24
3.3	Neural Network Toolbox	25
3.3.1	Open Neural Network Toolbox (NNTool)	25
3.3.2	Create Input, Target and Network	26
3.3.3	Train Network	29
3.4	GUI in GUIDE	30
3.4.1	Open GUIDE	30
3.4.2	Develop a GUI	31
3.4.3	Programming the GUI	35
4	RESULTS	38
4.1	Introduction	38
4.2	GUIDE Main Page Display	38
4.3	Recognition Results	39
4.3.1	Fingerprint Type A	39
4.3.2	Fingerprint Type B	41
4.3.3	Fingerprint Type C	42

4.3.4	Fingerprint Type D	44
4.3.5	Fingerprint Type E	45
4.3.6	Unrecognized Fingerprint	47
4.4	Close System	48
4.5	Discussion	49
5	CONCLUSION AND RECOMMENDATION	50
5.1	Introduction	50
5.2	Conclusion	50
5.3	Recommendation	51
	REFERENCES	52
	Appendices A-C	53-79

LIST OF TABLES

TABLE NO.	TITLE	PAGE
3.1	Features extract 1	24
3.2	Features extract 2	25

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	Types of Fingerprint	7
1.2	Parents Disciplines of AI	9
1.3	Simple Perceptron Layer	10
1.4	3 Perceptron Layers	11
1.5	Multilayer Perceptron	12
1.6	Project Operation Flowchart	18
1.7	Flowchart of System Procedure	19
2.1	Modeling Box for Neural Network	21
2.2	Neural Network Layers	22
2.3	Neural Network Concepts	22
3.1	Starting the NNTool	25
3.2	Network/Data Manager Explorer Window	26
3.3	Create the Data	27
3.4	Create the Network	28
3.5	Network/Data Manager after Setting the Data	28
3.6	Training the Data	29
3.7	Open the GUIDE	30
3.8	GUIDE Quick Start	31
3.9	Insert Push Button	32
3.10	Change Name of Push Button	33
3.11	Creating Axes	33
3.12	Saving Project GUI	34
3.13	Directory Save File Name	34
3.14	Function of Callback	35

3.15	Link to NNTool	36
3.16	Produce an Output with Defining the Variable Outputs	36
3.17	Import Image to the Axes	37
3.18	Creating an Exit Button	37
4.1	GUIDE Main Page	39
4.2	Performance of Type A	40
4.3	Output Display for Type A	40
4.4	Performance of Type B	41
4.5	Output Display for Type B	41
4.6	Performance of Type C	42
4.7	Output Display for Type C	43
4.8	Performance of Type D	44
4.9	Output Display for Type D	44
4.10	Performance of Type E	45
4.11	Output Display for Type E	46
4.12	Performance of Unrecognized Type	47
4.13	Output Display for Unrecognized Type	47
4.14	Exit Button Display	48

LIST OF SYMBOLS

pR	-	Input data
Wp	-	Weight
b	-	Bias
a_n	-	Output
Sn	-	Size of matrix

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligent
ANN	-	Artificial Neural Network
GUI	-	Graphical User Interface
GUIDE	-	Graphical User Interface Develop Environment
NNTool	-	Neural Network Toolbox
MLP	-	Multilayer Perceptron

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Flowchart of the GUI and Neural Network	
	Toolbox	53
B	Image of GUI Application	56
C	Programming of Graphical User Interface (GUI)	63

CHAPTER 1

INTRODUCTION

1.1 Fingerprint

In the 90 years since fingerprinting was generally introduced, out of the millions of sets of prints that have been taken, no two individuals have been found to have the same fingerprints. It is not the shape of the print that is individual, but rather the number, location and shape of specific ridge characteristics (also known as minutiae).

A fingerprint is an impression of the friction ridges of all or any part of the finger. A friction ridge is a raised portion of the epidermis on the palmar (palm and fingers) or plantar (sole and toes) skin, consisting of one or more connected ridge units of friction ridge skin. These ridges are sometimes known as "dermal ridges" or "dermal papillae".

Fingerprints may be deposited in natural secretions from the eccrine glands present in friction ridge skin (secretions consisting primarily of water) or they may be made by ink or other contaminants transferred from the peaks of friction skin ridges to a relatively smooth surface such as a fingerprint card. The term fingerprint normally refers to impressions transferred from the pad on the last joint of fingers and thumbs, though fingerprint cards also typically record portions of lower joint areas of the fingers (which are also used to make identifications).

1.2 Artificial Intelligent (AI)

The modern definition of artificial intelligence (or AI) is "the study and design of intelligent agents" where an intelligent agent is a system that perceives its environment and takes actions which maximizes its chances of success. John McCarthy, who coined the term in 1956, defines it as "the science and engineering of making intelligent machines." Other names for the field have been proposed, such as computational intelligence, synthetic intelligence or computational rationality. The term artificial intelligence is also used to describe a property of machines or programs: the intelligence that the system demonstrates.

1.3 Neural Network

A neural network, also known as a parallel distributed processing network, is a computing solution that is loosely modeled after cortical structures of the brain. It consists of interconnected processing elements called nodes or neurons that work together to produce an output function. The output of a neural network relies on the cooperation of the individual neurons within the network to operate. Processing of information by neural networks is characteristically done in parallel rather than in series (or sequentially) as in earlier binary computers or Von Neumann machines.

Neural network theory is sometimes used to refer to a branch of computational science that uses neural networks as models to simulate or analyze complex phenomena and/or study the principles of operation of neural networks analytically. It addresses problems similar to artificial intelligence (AI) except that AI uses traditional computational algorithms to solve problems whereas neural networks use 'networks of agents' (software or hardware entities linked together) as the computational architecture to solve problems. Neural networks are trainable systems that can "learn" to solve complex problems from a set of exemplars and generalize the "acquired knowledge" to solve unforeseen problems as in stock market

and environmental prediction. I.e., they are self-adaptive systems. The term 'Neural Network' has two distinct connotations:

- i. Biological neural networks are made up of real biological neurons that are connected or functionally-related in the peripheral nervous system or the central nervous system. In the field of neuroscience, they are often identified as groups of neurons that perform a specific physiological function in laboratory analysis.
- ii. Artificial neural networks are made up of interconnecting artificial neurons (usually simplified neurons) designed to model (or mimic) some properties of biological neural networks. Artificial neural networks can be used to model the modes of operation of biological neural networks, whereas cognitive models are theoretical models that mimic cognitive brain functions without necessarily using neural networks while artificial intelligence are well-crafted algorithms that solve specific intelligent problems (such as chess playing, pattern recognition, etc.) without using neural network as the computational architecture.

An artificial neural network (ANN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical model or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network.

Conventional approaches have been proposed for solving these problems. Although successful applications can be found in certain well-constrained environments, none is flexible enough to perform well outside its domain. ANNs provide exciting alternatives, and many applications could benefit from using them.

1.4 Graphical User Interface (GUI)

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders.

In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components. The applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work.

1.5 Problem Statement

In a common style, this will be a difficulty to know and differentiate the types of fingerprint. In this world, there are many types of marble and have a little different among them in the decorative design. We could not able to know the type of fingerprint which have the feature had been needed.

By using ANN toolbox, we able to assign an input pattern or train the network. This ability of the network is to recognize the data features that were extracted from the image with a little differentiate. Adding with using Graphical User Interface (GUI) almost inside the MATLAB is to be user friendly when using this system.

1.6 Objectives

The aim of this project is to design and implement a pattern recognition system that can recognize the type of fingerprint using MATLAB.

The main objectives of this project are:

- i. To recognize fingerprint image from the features extracted data for the recognition analysis using Artificial Neural Network (ANN).
- ii. To design and implement a pattern recognition system that can recognize the type of fingerprint based on the features extracted from the image.

1.7 Scope of Project

This project is to design and implement a pattern recognition system by using a graphical user interface (GUI) called NNTOOL. In this project, there is main target to achieve at the end of this project:

- i. To design and train the network in learning algorithm and weight initialization.
- ii. To study about using graphical user interface (GUI) include inside Neural Network toolbox.
- iii. To adapt of using the features extracted from the image which compare and produce a description about the image.
- iv. To use Graphical User Interface Develop Environment (GUIDE) to create the Graphical User Interface (GUI).

1.8 Literature review

1.8.1 Fingerprint

Fingerprint identification (sometimes referred to as dactyloscopy) or palmprint identification is the process of comparing questioned and known friction skin ridge impressions (see Minutiae) from fingers or palms to determine if the impressions are from the same finger or palm. The flexibility of friction ridge skin means that no two finger or palm prints are ever exactly alike (never identical in every detail), even two impressions recorded immediately after each other. Fingerprint identification (also referred to as individualization) occurs when an expert (or an expert computer system operating under threshold scoring rules) determines that two friction ridge impressions originated from the same finger or palm (or toe, sole) to the exclusion of all others. [1]

Fingerprint matching techniques can be placed into two categories: minutiae-based and correlation based. Minutiae-based techniques first find minutiae points and then map their relative placement on the finger. However, there are some difficulties when using this approach. It is difficult to extract the minutiae points accurately when the fingerprint is of low quality. Also this method does not take into account the global pattern of ridges and furrows. The correlation-based method is able to overcome some of the difficulties of the minutiae-based approach. However, it has some of its own shortcomings. Correlation-based techniques require the precise location of a registration point and are affected by image translation and rotation. [2]

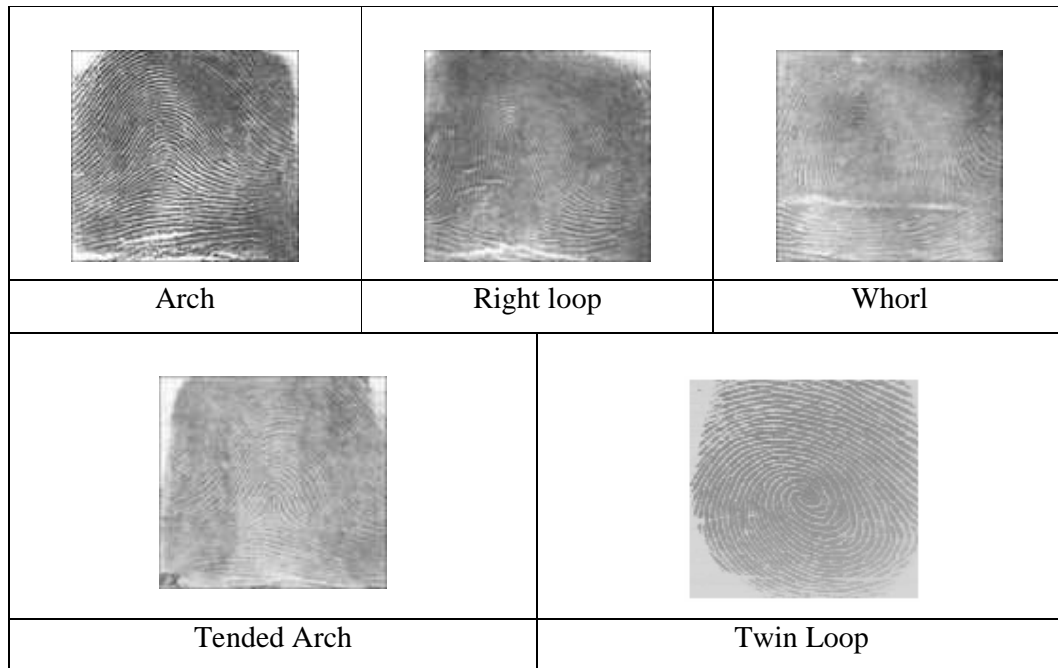


Figure 1.1 Types of Fingerprint [3]

1.8.2 Artificial Intelligent (AI)

Conventional AI mostly involves methods now classified as machine learning, characterized by formalism and statistical analysis. This is also known as symbolic AI, logical AI, neat AI and Good Old Fashioned Artificial Intelligence (GOFAI). Methods include:

- i. Expert systems: apply reasoning capabilities to reach a conclusion. An expert system can process large amounts of known information and provide conclusions based on them.
- ii. Case based reasoning: stores a set of problems and answers in an organized data structure called cases. A case based reasoning system upon being presented with a problem finds a case in its knowledge base that is most closely related to the new problem and presents its solutions as an output with suitable modifications.

- iii. Bayesian networks
- iv. Behavior based AI: a modular method of building AI systems by hand.

Conventional AI research focuses on attempts to mimic human intelligence through symbol manipulation and symbolically structured knowledge bases. This approach limits the situations to which conventional AI can be applied. Lotfi Zadeh stated that "we are also in possession of computational tools which are far more effective in the conception and design of intelligent systems than the predicate-logic-based methods which form the core of traditional AI." These techniques, which include fuzzy logic, have become known as soft computing. These often biologically inspired methods stand in contrast to conventional AI and compensate for the shortcomings of symbolicism. These two methodologies have also been labeled as neats vs. scruffies, with neats emphasizing the use of logic and formal representation of knowledge while scruffies take an application-oriented heuristic bottom-up approach. [4]

The subject of AI spans a wide horizon. It deals with the various kinds of knowledge representation schemes, different techniques of intelligent search, various methods for resolving uncertainty of data and knowledge, different schemes for automated machine learning and many others. Among the application areas of AI, we have Expert systems, Game-playing, and Theorem-proving, Natural language processing, Image recognition, Robotics and many others. The subject of AI has been enriched with a wide discipline of knowledge from Philosophy, Psychology, Cognitive Science, Computer Science, Mathematics and Engineering. Thus in Figure 1.2, they have been referred to as the parent disciplines of AI. An at-a-glance look at Figure 1.2 also reveals the subject area of AI and its application areas. [5]

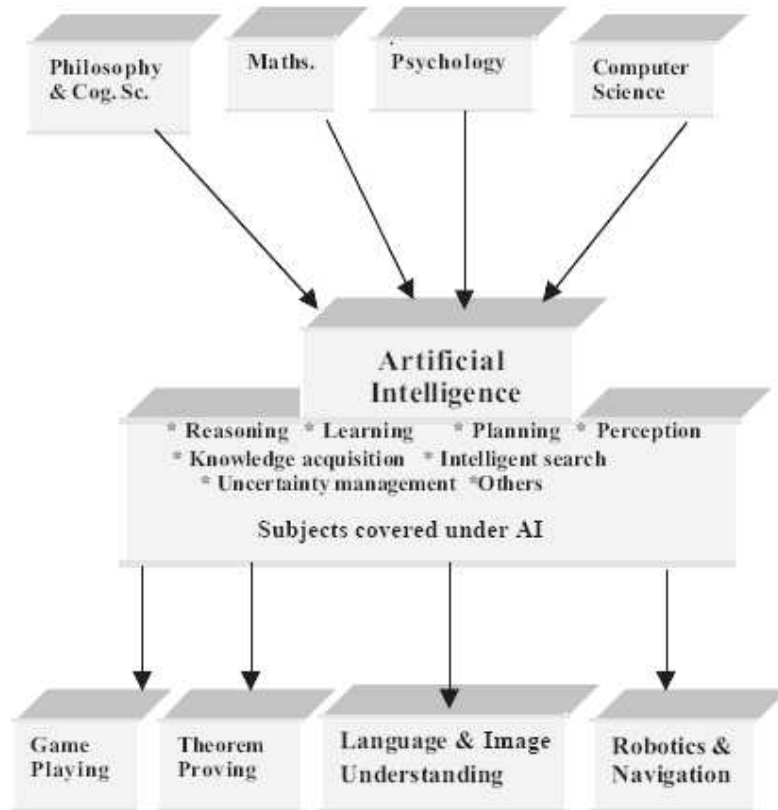


Figure 1.2 Parent Disciplines of AI [5]

1.8.3 Neural Network Toolbox

The first important part is using Neural Network toolbox. The Neural Network toolbox is a collection of function built in MATLAB which allows you to create and edit fuzzy inference system within the framework of MATLAB environment and integrate into simulations using SIMULINK. [1]

A neuron with a single R-element input vector is shown below. Here the individual element inputs

$$p_1, p_2, \dots, p_R$$

are multiplied by weights

$$w_{1,1}, w_{1,2}, \dots, w_{1,R}$$

and the weighted values are fed to the summing junction. Their sum is simply \mathbf{Wp} , the dot product of the (single row) matrix \mathbf{W} and the vector \mathbf{p} .

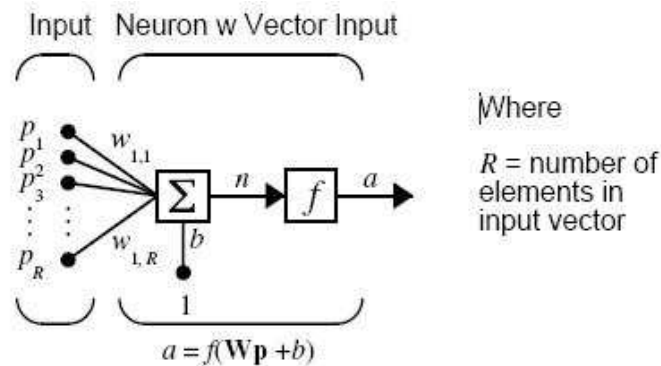


Figure 1.3 Simple Perceptron Layer

The neuron has a bias b , which is summed with the weighted inputs to form the net input n . This sum, n , is the argument of the transfer function f .

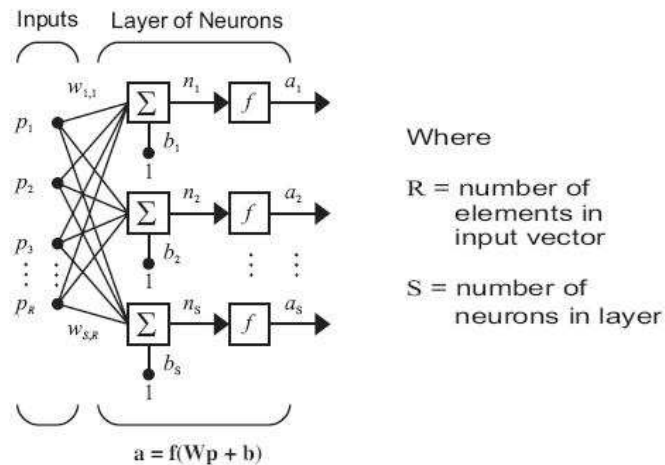
$$n = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b$$

This expression can, of course, be written in MATLAB code as

$$n = \mathbf{W} * \mathbf{p} + b$$

However, you will seldom be writing code at this level, for such code is already built into functions to define and simulate entire networks.

Two or more of the neurons shown earlier can be combined in a layer, and a particular network could contain one or more such layers. First consider a single layer of neurons. A one-layer network with R input elements and S neurons follows.



Where
 R = number of elements in input vector
 S = number of neurons in layer

Figure 1.4 3 Perceptron Layers

In this network, each element of the input vector \mathbf{p} is connected to each neuron input through the weight matrix \mathbf{W} . The i th neuron has a summer that gathers its weighted inputs and bias to form its own scalar output $n(i)$. The various $n(i)$ taken together form an S -element net input vector \mathbf{n} . Finally, the neuron layer outputs form a column vector \mathbf{a} . The expression for \mathbf{a} is shown at the bottom of the figure.

Note that it is common for the number of inputs to a layer to be different from the number of neurons (i.e., R is not necessarily equal to S). A layer is not constrained to have the number of its inputs equal to the number of its neurons.

You can create a single (composite) layer of neurons having different transfer functions simply by putting two of the networks shown earlier in parallel. Both networks would have the same inputs, and each network would create some of the outputs. The input vector elements enter the network through the weight matrix \mathbf{W} .

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

Note that the row indices on the elements of matrix \mathbf{W} indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, the indices in $w_{1,2}$ say that the strength of the signal from the second input element to the first (and only) neuron is $w_{1,2}$. The S neuron R input one-layer network also can be drawn in abbreviated notation.

A network can have several layers. Each layer has a weight matrix \mathbf{W} , a bias vector \mathbf{b} , and an output vector \mathbf{a} . To distinguish between the weight matrices, output vectors, etc., for each of these layers in the figures, the number of the layer is appended as a superscript to the variable of interest. You can see the use of this layer notation in the three-layer network shown below, and in the equations at the bottom of the figure.

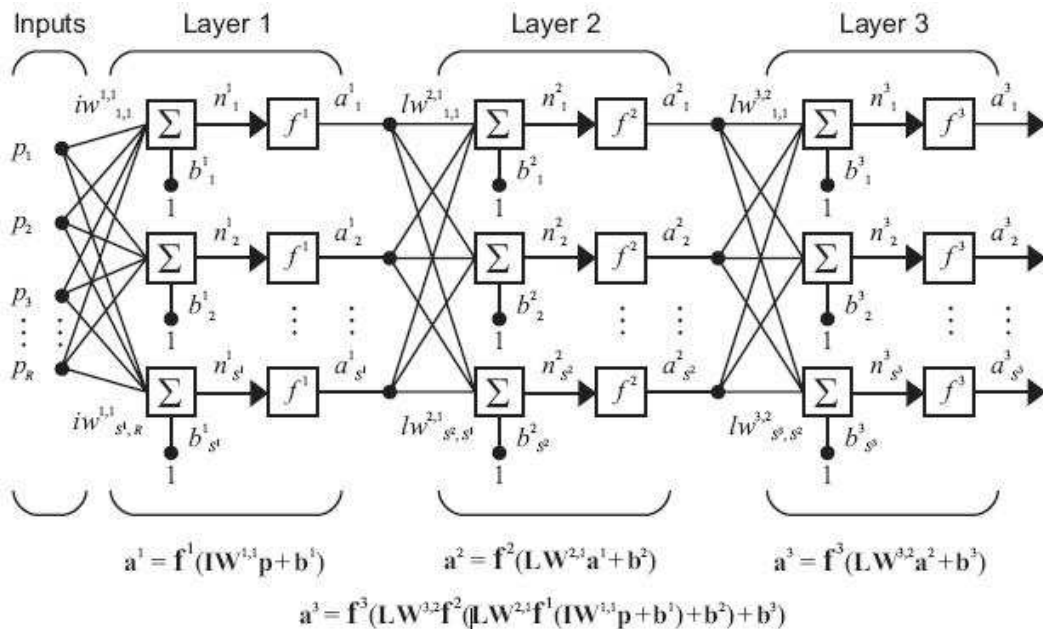


Figure 1.5 Multilayer Perceptron

The network shown above has $R1$ inputs, $S1$ neurons in the first layer, $S2$ neurons in the second layer, etc. It is common for different layers to have different numbers of neurons. A constant input 1 is fed to the bias for each neuron.

Note that the outputs of each intermediate layer are the inputs to the following layer. Thus layer 2 can be analyzed as a one-layer network with $S1$ inputs, $S2$ neurons, and an $S2 \times S1$ weight matrix $W2$. The input to layer 2 is $a1$; the output is $a2$. Now that all the vectors and matrices of layer 2 have been identified, it can be treated as a single-layer network on its own. This approach can be taken with any layer of the network. The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers. The three-layer network shown earlier has one output layer (layer 3) and two hidden layers (layer 1 and layer 2). Some authors refer to the inputs as a fourth layer. This toolbox does not use that designation. [6]

The principles behind the toolbox are more important than simply compiling lists of algorithms. Data analysis and modeling methods should not be used in isolation; all parts of the toolbox interact in a coherent way, and implementations of standard pattern recognition techniques (such as linear regression and K-nearest-neighbour classifiers) are provided so that they can be used as benchmarks against which more complex algorithms can be evaluated. This interaction allows researchers to develop new techniques by building on and reusing existing software, thus reducing the effort required and increasing the robustness and usability of the new tools.

Supervised neural networks are trained to produce desired outputs in response to sample inputs, making them particularly well suited to modeling and controlling dynamic systems, classifying noisy data, and predicting future events.

- i. Feedforward networks have one-way connections from input to output layers. They are most commonly used for prediction, pattern recognition, and nonlinear function fitting. Supported Feedforward

networks include Feedforward Backpropagation, Cascade-forward Backpropagation, Feedforward Input-delay Backpropagation, Linear, and Perceptron networks.

- ii. Radial Basis networks provide an alternative, fast method for designing nonlinear Feedforward networks. Supported variations include generalized regression and probabilistic Neural Networks.
- iii. Dynamic networks use memory and recurrent feedback connections to recognize spatial and temporal patterns in data. They are commonly used for time-series prediction, nonlinear dynamic system modeling, and control system applications. Prebuilt dynamic networks in the toolbox include focused and distributed time-delay, nonlinear autoregressive (NARX), layer-recurrent, Elman, and Hopfield networks. The toolbox also supports dynamic training of custom networks with arbitrary connections.

Unsupervised neural networks are trained by letting the network continually adjust itself to new inputs. They find relationships within data and can automatically define classification schemes.

The Neural Network Toolbox supports two types of self-organizing, unsupervised networks: competitive layers and self-organizing maps. Competitive layers recognize and group similar input vectors. By using these groups, the network automatically sorts the inputs into categories.

Self-organizing maps learn to classify input vectors according to similarity. Unlike competitive layers they also preserve the topology of the input vectors, assigning nearby inputs to nearby categories. [7]

1.8.4 Pattern Recognition

Pattern recognition is a sub-topic of machine learning. It can be defined as "The act of taking in raw data and taking an action based on the category of the data"

Most research in pattern recognition is about methods for supervised learning and unsupervised learning.

Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.

A complete pattern recognition system consists of a sensor that gathers the observations to be classified or described; a feature extraction mechanism that computes numeric or symbolic information from the observations; and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features.

The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set and the resulting learning strategy is characterized as supervised learning. Learning can also be unsupervised, in the sense that the system is not given an a priori labeling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches: statistical (or decision theoretic), syntactic (or structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Structural pattern recognition is based on the structural interrelationships of features. A wide range of algorithms can

be applied for pattern recognition, from very simple Bayesian classifiers to much more powerful neural networks.

Pattern recognition is more complex when templates are used to generate variants. For example, in English, sentences often follow the "N-VP" (noun - verb phrase) pattern, but some knowledge of the English language is required to detect the pattern. Pattern recognition is studied in many fields, including psychology, ethology, and computer science. [8]

1.8.5 Graphical User Interface (GUI) in GUIDE

GUI design is an important adjunct to application programming. Its goal is to enhance the usability of the underlying logical design of a stored program. The visible graphical interface features of an application are sometimes referred to as "chrome". They include graphical elements (widgets) that may be used to interact with the program. Common widgets are: windows, buttons, menus, and scroll bars. Larger widgets, such as windows, usually provide a frame or container for the main presentation content such as a web page, email message or drawing. Smaller ones usually act as a user-input tool.

The widgets of a well-designed system are functionally independent from and indirectly linked to program functionality, so the GUI can be easily customized, allowing the user to select or design a different skin at will. [9]

A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use. For example, it is much easier for a new user to move a file from one directory to another by dragging its icon with the mouse than by having to remember and type seemingly arcane commands to accomplish the same task.

Adding to this intuitiveness of operation is the fact that GUIs generally provide users with immediate, visual feedback about the effect of each action. For example, when a user deletes an icon representing a file, the icon immediately disappears, confirming that the file has been deleted (or at least sent to the trash can). This contrast with the situation for a CLI, in which the user types a delete command (inclusive of the name of the file to be deleted) but receives no automatic feedback indicating that the file has actually been removed.

In addition, GUIs allow users to take full advantage of the powerful multitasking (the ability for multiple programs and/or multiple instances of single programs to run simultaneously) capabilities of modern operating systems by allowing such multiple programs and/or instances to be displayed simultaneously. The result is a large increase in the flexibility of computer use and a consequent rise in user productivity.

But the GUI has become much more than a mere convenience. It has also become the standard in human-computer interaction, and it has influenced the work of a generation of computer users. Moreover, it has led to the development of new types of applications and entire new industries. An example is desktop publishing, which has revolutionized (and partly wiped out) the traditional printing and typesetting industry. [10]

1.9 Methodology

1.9.1 Introduction

In doing a successful project, a methodology is one of the important elements. Methodology is the set of procedure of the project flow which includes the theories, concepts or ideas, comparative study of different approaches and critique of the individual methods will make sure that the project will run smoothly according to plan and to make sure that we can get the expected result.

1.9.2 Methodology

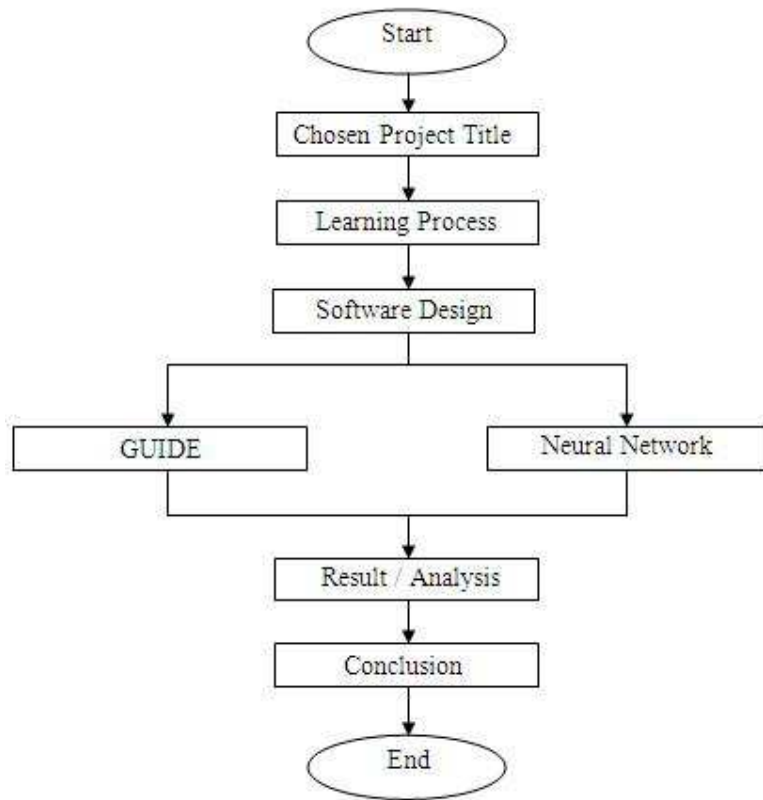


Figure 1.6 Project Operation Flowchart

1.9.3 Software Design

To be done this project, the Neural Network Toolbox is used to accomplish the scope of the project. The additional software is used such as Graphical User Interface (GUI) in GUIDE for make a system is well designed and easy to use. This also called as “user friendly”.

1.9.4 Software Procedure

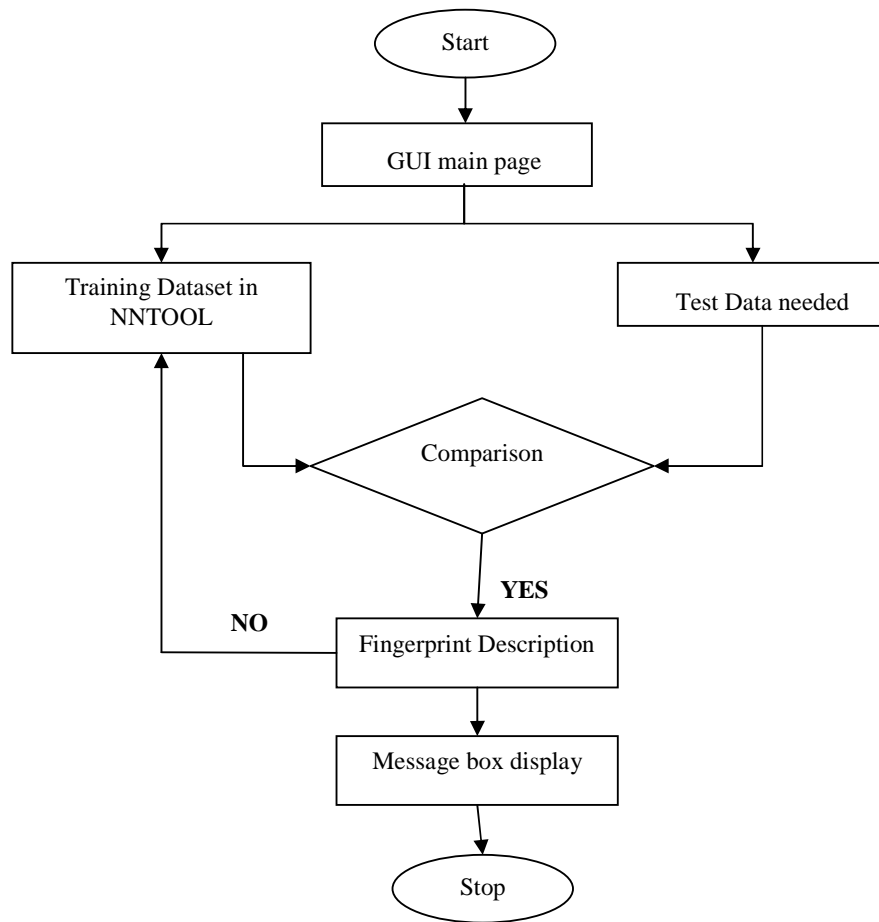


Figure 1.7 Flowchart of System Procedure

After developing the Neural Network Toolbox which is the main part of this system, the Graphical User Interface (GUI) in GUIDE is used to make the system is more systematic and user easy to evaluate the result just only insert data and clicking the buttons. The output result will appear after simulating the system application.

1.9.5 Result and Analysis

From the result of the system, data and output will be compared. M-file which is constructed inside GUI is used to define the output in the statement that easy to understand.

1.9.6 Conclusion

After the system is developed, and description of the fingerprint is carried out, a conclusion is made in order to see the successfulness of this project based on the objectives that were set earlier. Thus, a recommendation is made for future progress for enhancement of this system.

CHAPTER 2

SYSTEM MODEL

2.1 Introduction

In order to design this project, a system modeling is necessary to provide method for the control system. By this, a descriptive model of the system as a hypothesis of how the system could work is built. For the neural network, the system is able to be constructing by weight and bias. This is the most important thing.

2.2 System Model



Figure 2.1 Modeling Box for Neural Network

From the Black-Box above, there is no previous knowledge, but there are measurements, observations, records, and data.

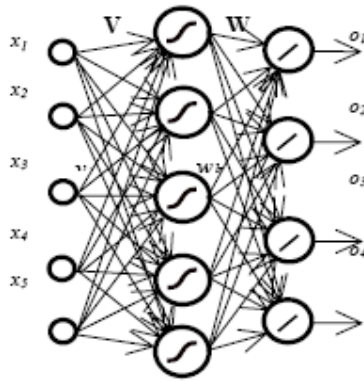


Figure 2.2 Neural Network Layers

As we know, inside a single box from the Figure 2.1 above, there stands the idea of learning from the data behind the Neural Network. If there do not have any prior knowledge AND there do not have any measurements (by all accounts very hopeless situation indeed) it may be hard to expect or believe that the problem at hand may be approached and solved easily.

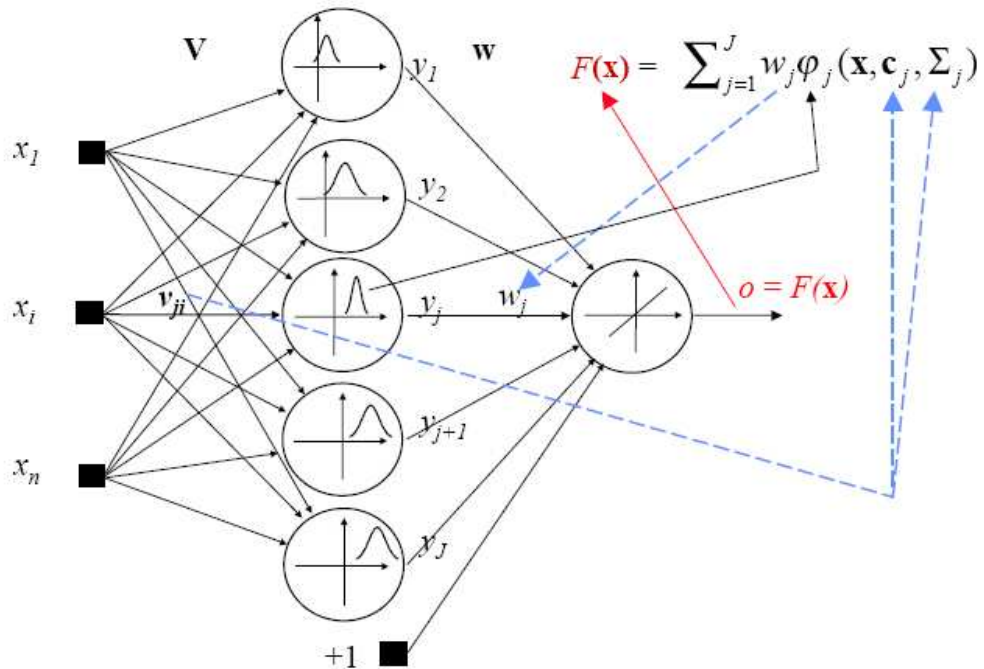


Figure 2.3 Neural Network Concepts

From the Figure 2.3 above, there are equations constructed inside the Neural Network. The bias and weight are affecting the output of the data. Error of the output data will be reanalyzed until the needed data is reached. In reaching that, the weight and bias must be reinitializing by supervised learning or unsupervised learning.

- i. Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.
- ii. In the most of real-life problems, a stochastic component of data is the normal probability distribution law, i.e., the underlying joint probability distribution is Gaussian.
- iii. Due to the second assumption, the induction paradigm for parameter estimation is the maximum likelihood method that is reduced to the minimization of the sum-of-errors-squares cost function in most engineering applications.

CHAPTER 3

SOFTWARE

3.1 Introduction

The Graphical User Interface (GUI) is designed to be simple and user friendly. A simple example will get you started. Inside the Neural Network Toolbox also have the GUI which called “NNTool”. This application still same as the previous Neural Network, but it is a simplification of using the command.

3.2 Training Data Table

Table 3.1 : Features extract 1

	Fore ground	Total Min	Min 06	Min 065	Min 075	Min 08	Min 09	Normed Match
Type A	0.101	0.189	0.290	0.302	0.516	0.302	0.302	0.287
Type B	0.054	0.102	0.038	0.382	0.448	0.160	0.159	0.276
Type C	0.604	0.485	0.504	0.506	0.513	0.449	0.448	0.298
Type D	0.072	0.310	0.558	0.561	0.560	0.478	0.478	0.323
Type E	0.120	0.071	0.385	0.686	0.608	0.240	0.239	0.436

Table 3.2 : Features extract 2

	Fore ground	Total Min	Min 06	Min 065	Min 075	Min 08	Min 09	Normed Match
Type A	0.166	0.330	0.403	0.597	0.274	0.424	0.424	0.411
Type B	0.042	0.089	0.028	0.333	0.412	0.156	0.155	0.272
Type C	0.452	0.333	0.376	0.562	0.520	0.425	0.424	0.274
Type D	0.081	0.321	0.565	0.542	0.562	0.451	0.451	0.296
Type E	0.127	0.078	0.292	0.718	0.589	0.303	0.302	0.499

3.3 Neural Network Toolbox

3.3.1 Open Neural Network Toolbox (NNTool)

In GUI, to create a perceptron network is to perform the AND function. It has an input vector ‘p’ and a target vector ‘t’. Just call the network as Network.

To start, type ‘nntool’ at the command window after open the MATLAB software. Other way to get the GUI of the Neural Network as follow the instruction (Start in MATLAB menu > Toolboxes > NNTool) or shown as the figure 8 below;

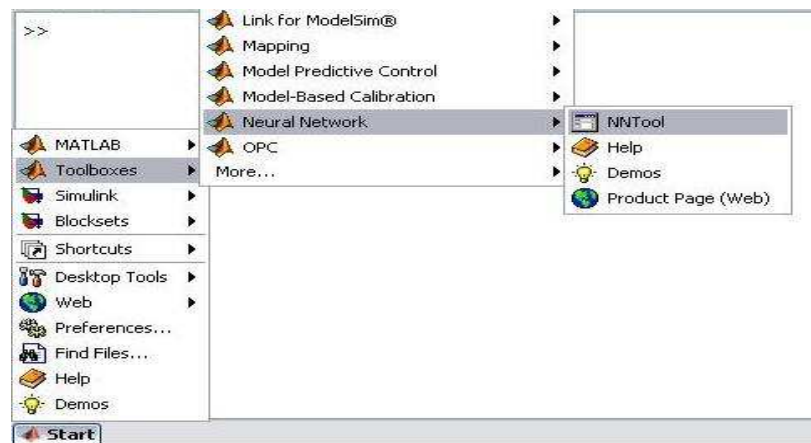


Figure 3.1 Starting the NNTool

The Network/Data Manager window will appear as shown in Figure 3.2;

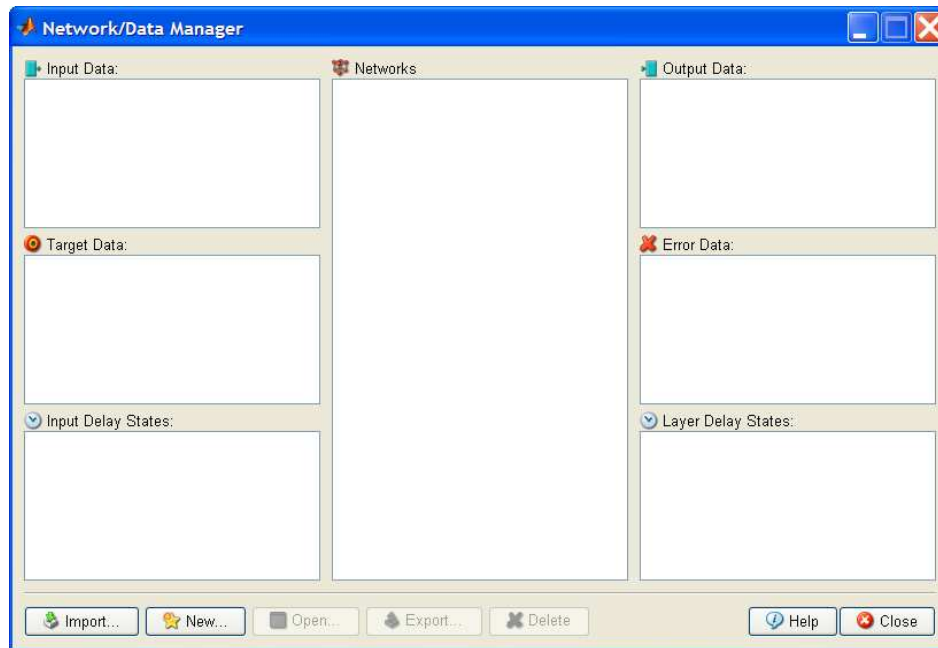


Figure 3.2 Network/Data Manager Explorer Window

3.3.2 Create Input, Target and Network

First, define the network input, called 'Input'. To define this data, from the Network/Data Manager as shown on Figure 3.2, click New, and a new window, Create Network or Data, appears. Select the Data tab. Set the Name to Input, the Value refer to the training data table, and make sure that Data Type is set to Inputs. Also the same type if we need to add target 't1' and we need to make sure the Data Type is set to Targets.

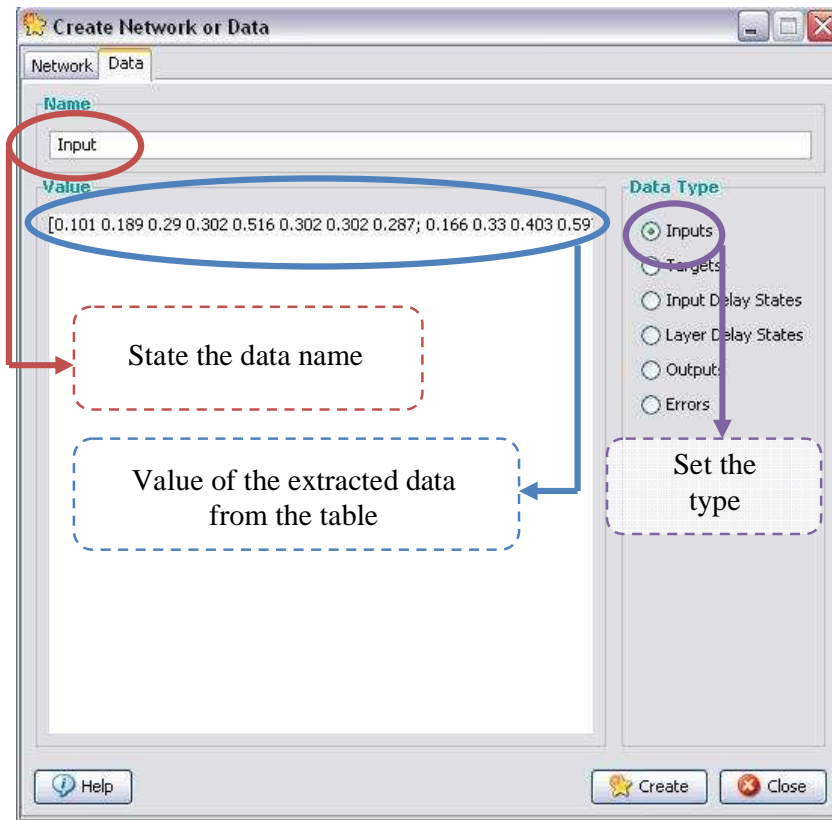


Figure 3.3 Create the Data

Create a new network and call it Fingerprint. Select the Network tab. Enter Network under Name. Set the Network Type to Perceptron, for that is the kind of network you want to create.

Set the input ranges by entering numbers in that field, but it is easier to get them from the particular input data that you want to use. To do this, click the down arrow at the right side of Input Range. This pull-down menu shows that you can get the input ranges from the file Input. That is what you want to do, so click Input.

You need to use a hardlim transfer function and a learnp learning function, so set those values using the arrows for Transfer function and Learning function, respectively. By now your Create Network or Data window should look like the Figure 3.4;

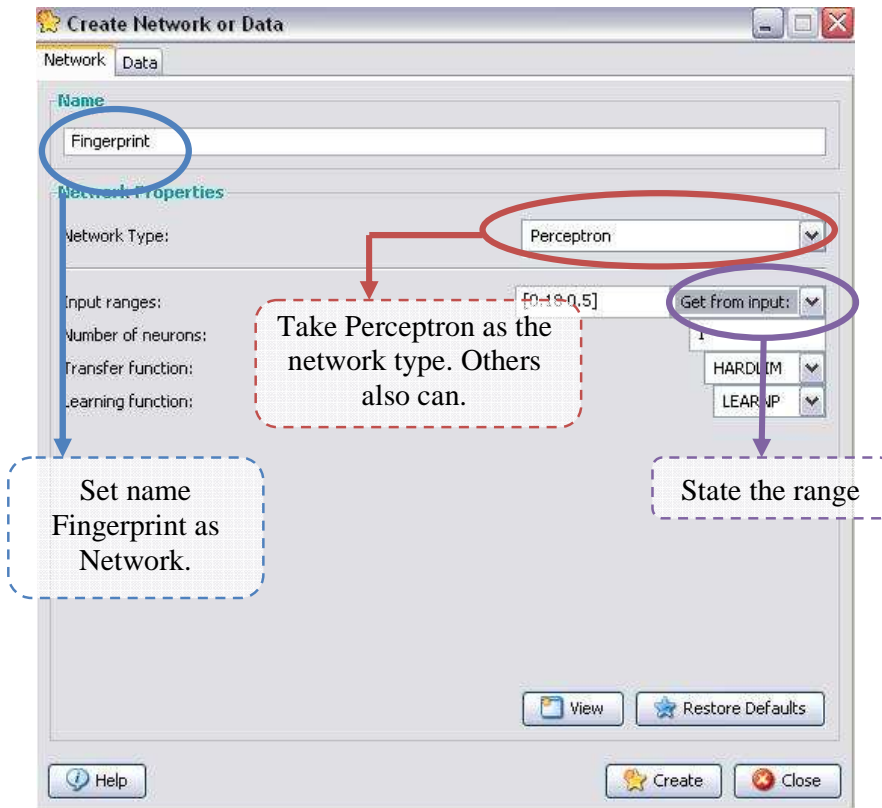


Figure 3.4 Create the Network

After setting the data, target and Network, at least will show as Figure 3.5 below;

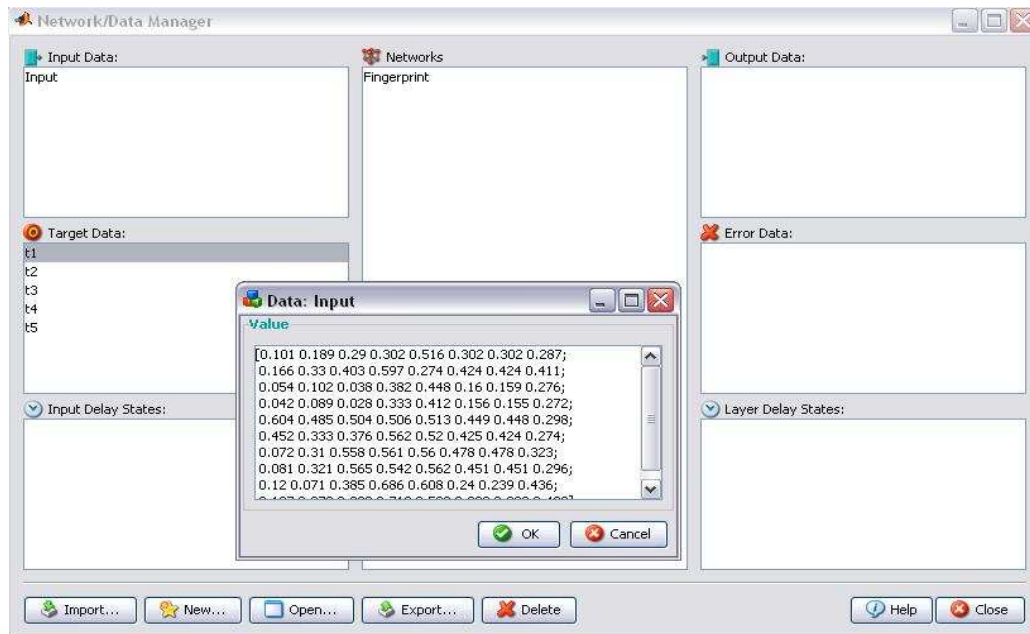


Figure 3.5 Network/Data Manager after Setting the Data

3.3.3 Train Network

To train the network, click Fingerprint to highlight it. Then click Open. This leads to a new window, labeled Network: Fingerprint. At this point you can see the network again by clicking the View tab. You can also check on the initialization by clicking the Initialize tab. Now click the Train tab. specify the inputs and output by clicking the Training Info tab and selecting “Input” from the list of inputs and “t1” from the list of targets.

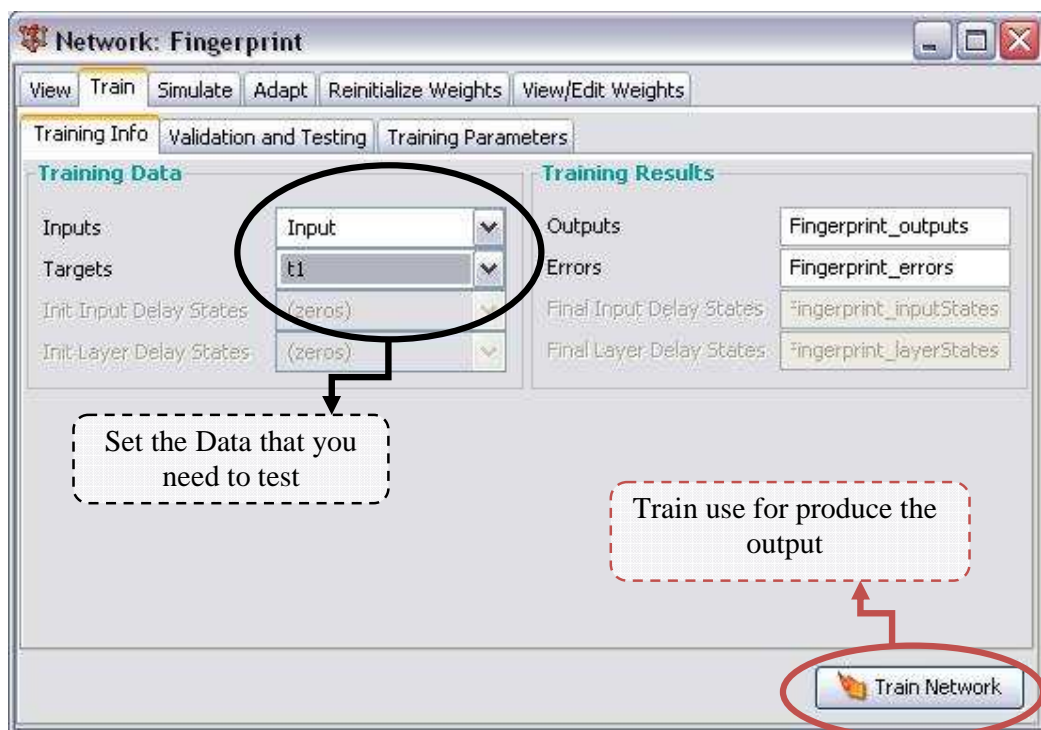


Figure 3.6 Training the Data

3.4 GUI in GUIDE

3.4.1 Open GUIDE

From MATLAB File menu, click New > GUI. From the MATLAB explorer, click the button at the menu bar as shown as Figure 3.7;

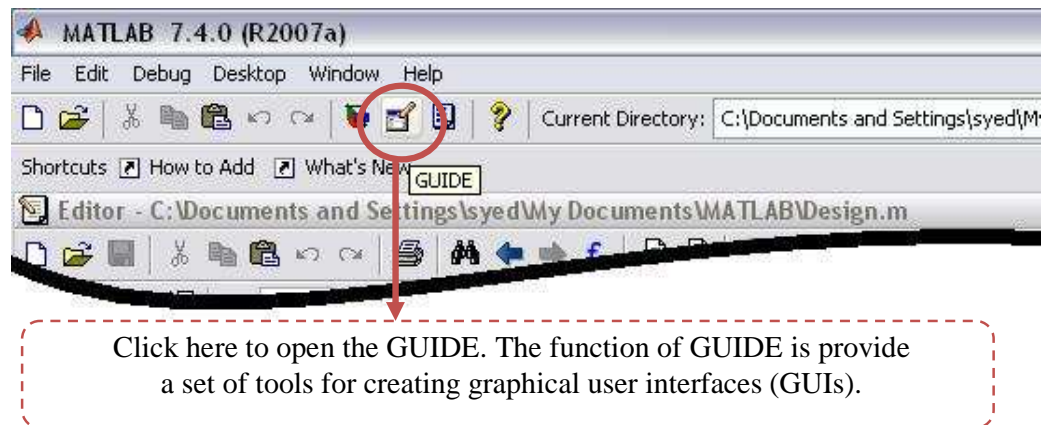


Figure 3.7 Open the GUIDE

The displays the GUIDE Quick Start dialog box shown in the following Figure 3.8. The GUIDE Quick Start dialog box contains two tabs:

- i. Create New GUI – Asks you to start creating your new GUI by choosing a template for it. You can also specify the name by which the GUI is saved.
- ii. Open Existing GUI – Enables you to open an existing GUI in GUIDE. You can choose a GUI from your current directory or browse other directories.

At GUIDE templates, chose the Blank GUI (Default), and click OK to display the blank templates. This is for user first time of making the GUI.

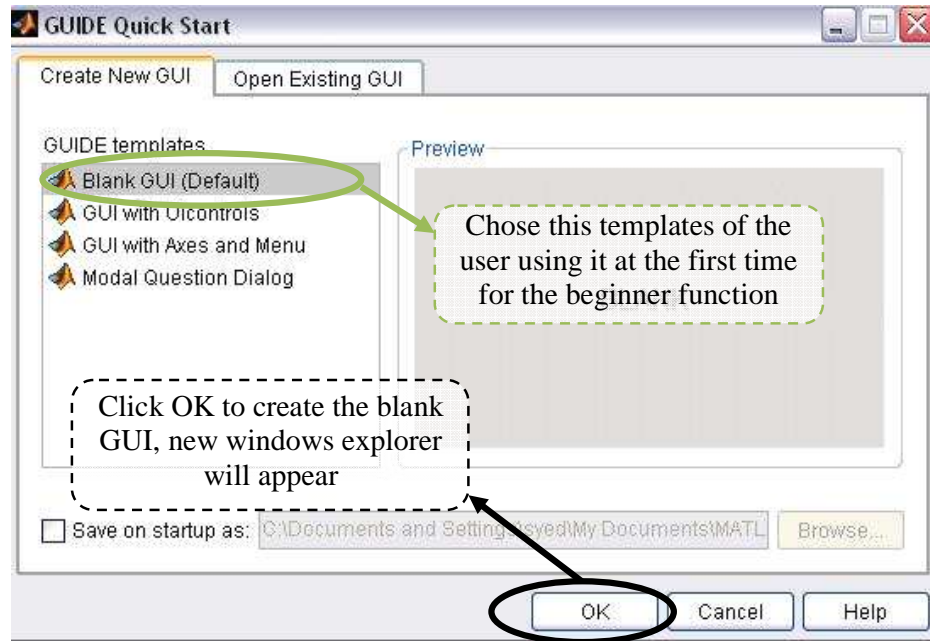


Figure 3.8 GUIDE Quick Start

3.4.2 Develop a GUI

After click OK to display the blank GUI in the Layout Editor, the Fig-file will appear as shown in the following Figure 3.9. Set the size of the GUI by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until fit to the size that the user needed. To create the push button, select the push button from the component palette at the left of the Layout Editor and drag it into the layout area as shown in the Figure 3.9;

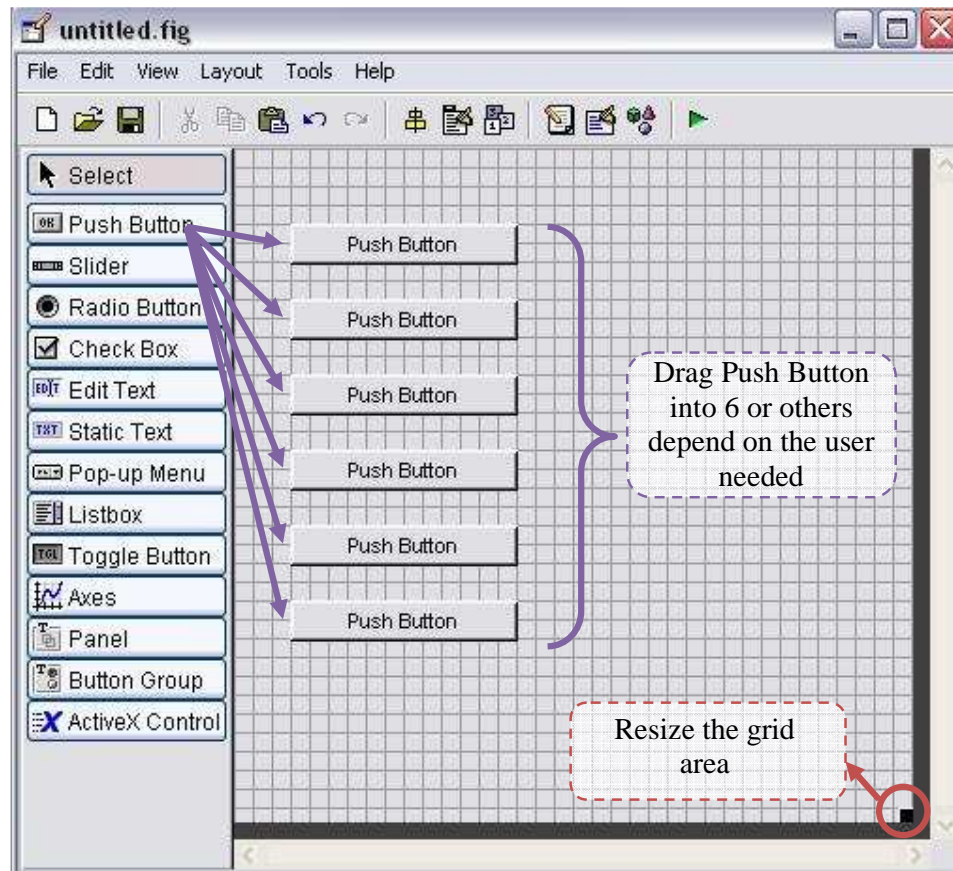


Figure 3.9 Insert Push Button

To change the name at the Push Button, right click and choose property inspector. The explorer will appear as shown as Figure 3.10. Then change the name at String from 'Push Button' to 'Introduction' as shown as Figure 3.10. For the others push button, use the same step which every tab have their own function.

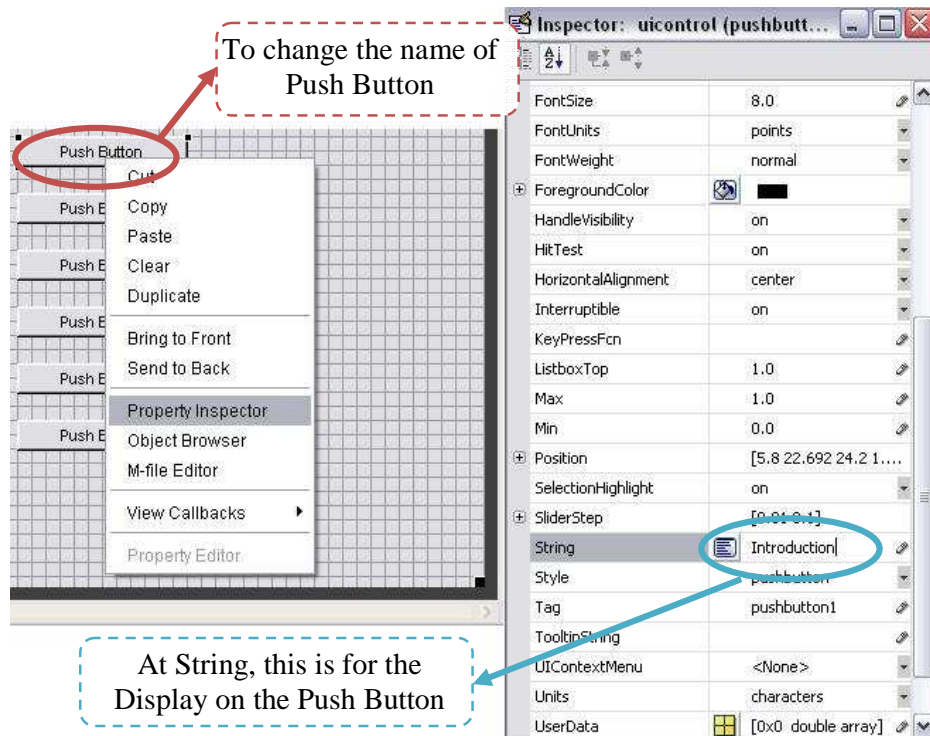


Figure 3.10 Change Name of Push Button

To insert a picture in GUI, click Axes and drag as shown as Figure 3.11. Axes are also function to display plotting graph from the operation had given.

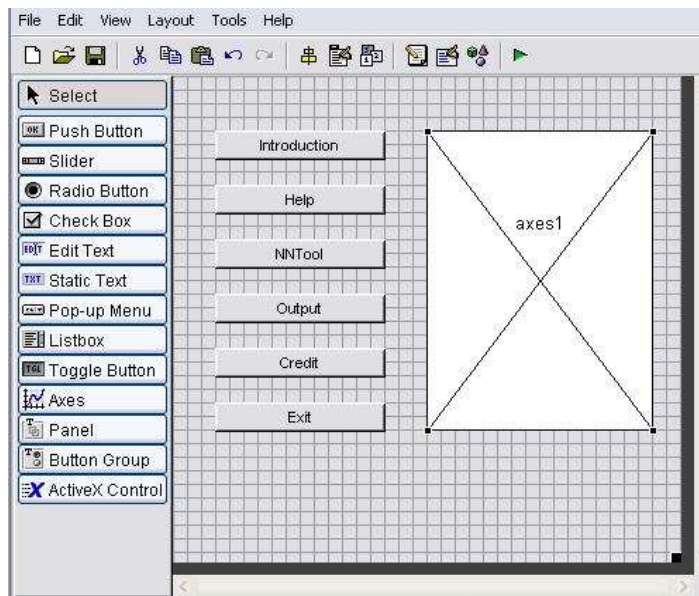


Figure 3.11 Creating Axes

When you save a GUI, GUIDE creates two files, a FIG-file and an M-file. The FIG-file, with extension .fig, is a binary file that contains a description of the layout. The M-file, with extension .m, contains the code that controls the GUI.

- i. Save and activate your GUI by selecting Run from the Tools menu.
- ii. GUIDE displays the following dialog box. Click Yes to continue.

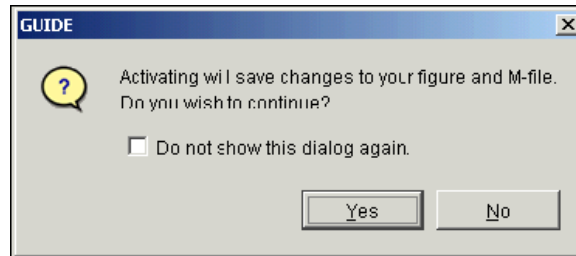


Figure 3.12 Saving Project GUI

- iii. GUIDE opens a Save As dialog box in your current directory and prompts you for a FIG-file name.

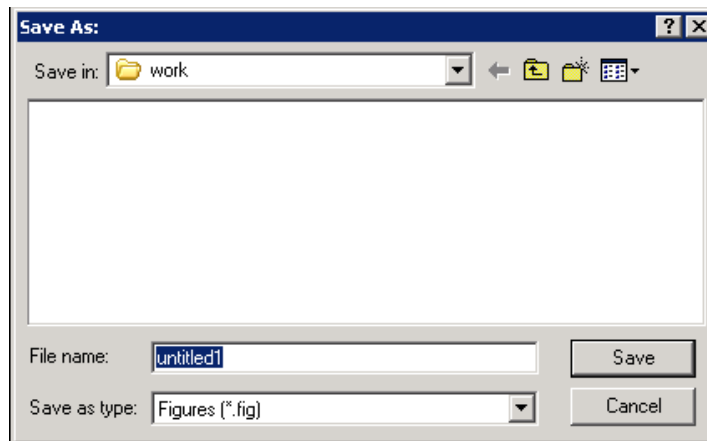


Figure 3.13 Directory Save File Name

- iv. Browse to any directory for which you have write privileges, and then enter the filename Print for the FIG-file. GUIDE saves both the FIG-file and the M-file using this name.

3.4.3 Programming the GUI

From the push button at the layout area, right click view callback > callback which is connect to the M-file that had been construct by saving the project as explain as before. Figure 3.14 shows prove of the function callback. With the command “figure (introduction)” means that this button is connect or link with other GUI template called ‘introduction’. These are the function of callback;

- i. Routine that executes whenever you activate the uicontrol object .
- ii. Define this routine as a string that is a valid MATLAB expression or the name of an M-file.
- iii. The expression executes in the MATLAB workspace.

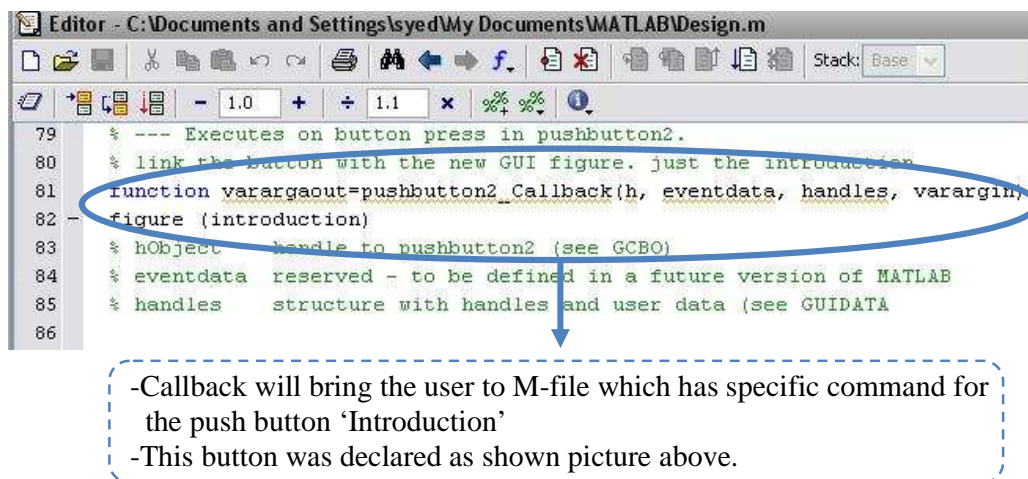


Figure 3.14 Function of Callback

For the link this GUI template with other application such as Neural Network Toolbox (NNTool), the command is shown as Figure 3.15.

```

96
97 % --- Executes on button press in pushbutton5.
98 % link the button with the other GUI concept in neural network toolbox. new
99 % windows of the GUI will appear, otherwise it also can be display with
100 % manually called.
101 function varargout=pushbutton5_Callback(h, eventdata, handles, varargin)
102 NNTool
103 % hObject handle to pushbutton5 (see GCBO)
104 % eventdata reserved - to be defined in a future version of MATLAB
105 % handles structure with handles and user data (see GUIDATA)
106

```

Just type NNTool which is to connect the Neural Network toolbox with the push button.

Figure 3.15 Link to NNTool

Every system has their own result whenever shown in graph, statement at command window or others. So, just creating a push button which is to simplify the user just click the button and result will appear. Example is this fingerprint recognition system. Figure 3.16 shows the command at the M-file also the function of the defining the output.

```

109 % link the button with the figure out the same function in window command.
110 function pushbutton6_Callback(hObject, eventdata, handles)
111 a=evalin('base','Fingerprint_outputs')
112
113 if a>=[0 1 0 0 1 0 1 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0]
114 message=strcat('The Fingerprint is type A');
115 msgbox(message);
116 elseif a>=[0 0 1 0 1 0 1 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0]
117 message=strcat('The Fingerprint is type B');
118

```

a=evalin ('*', '*')
 -'base' is to import data from workspace into function.
 -'Fingerprint_outputs' is the result which was imported by the function above

'a' is the defining for the equation.
 The value inside is one of the result.

Figure 3.16 Produce an Output with Defining the Variable Outputs

For the axes that want to insert picture, this apart of making GUI more valuable. Just copy the picture needed and insert at folder MATLAB which is the main path of reference to be search in. The Figure 3.17 shows that how to import image from the path and put it into axes.

```
61 % UIWAIT makes Design wait for user response (see UIRESUME)
62 % uiwait(handles.figure1);
63 % to make a background just only use axes by resize fit to the work size
64 [x,map]=imread('jejari','jpg');
65 image(x)
66 set(gca,'visible','off')
67
68 % --- Outputs from this function
69 function varargout = Design_Output
70 % varargout cell array for return
```

[x,map]=imread ('*', 'jpg') is to import image to the axes. 'jejari' is the image file name in format jpeg.

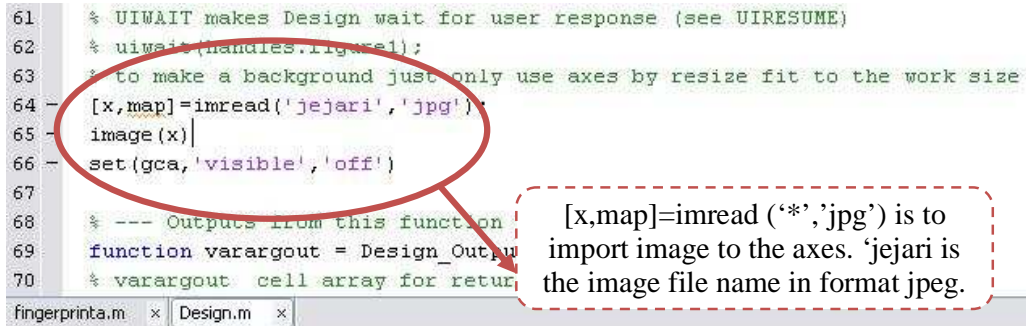


Figure 3.17 Import Image to the Axes

Creating an exit button with message box is the function of quit of the whole system. This is a part of completing the system. At the end in making GUIDE, save the entire project. After that run GUI template and see the result of each function.

```
169 - button = questdlg(' Are you sure to exit?', 'Yes', 'No', 'No');
170 - switch button
171 -     case 'Yes',
172 -         close all
173 -     case 'No',
174 -         quit cancel;
175 - end
```

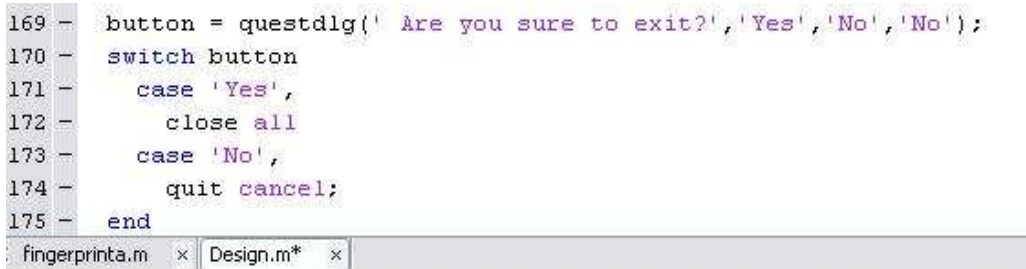


Figure 3.18 Creating an Exit Button

CHAPTER 4

RESULTS

4.1 Introduction

As the results, this system is able to recognize the type of the fingerprint and produce and output at the matrix form. In this system, it is able to define the five types of the fingerprint and will display others if the data is not equal to the training data. With using GUIDE of making GUI, it is able to make user easy to use the system. With creating define function inside M-file, it able to produce a display about the fingerprint recognizing.

4.2 GUIDE Main Page Display

The GUI of GUIDE is used as the main page of the whole system. Every button has its own link to other application and description. To further this recognition system, click the button 'Go to NNTool' and insert data as explain in past chapter.

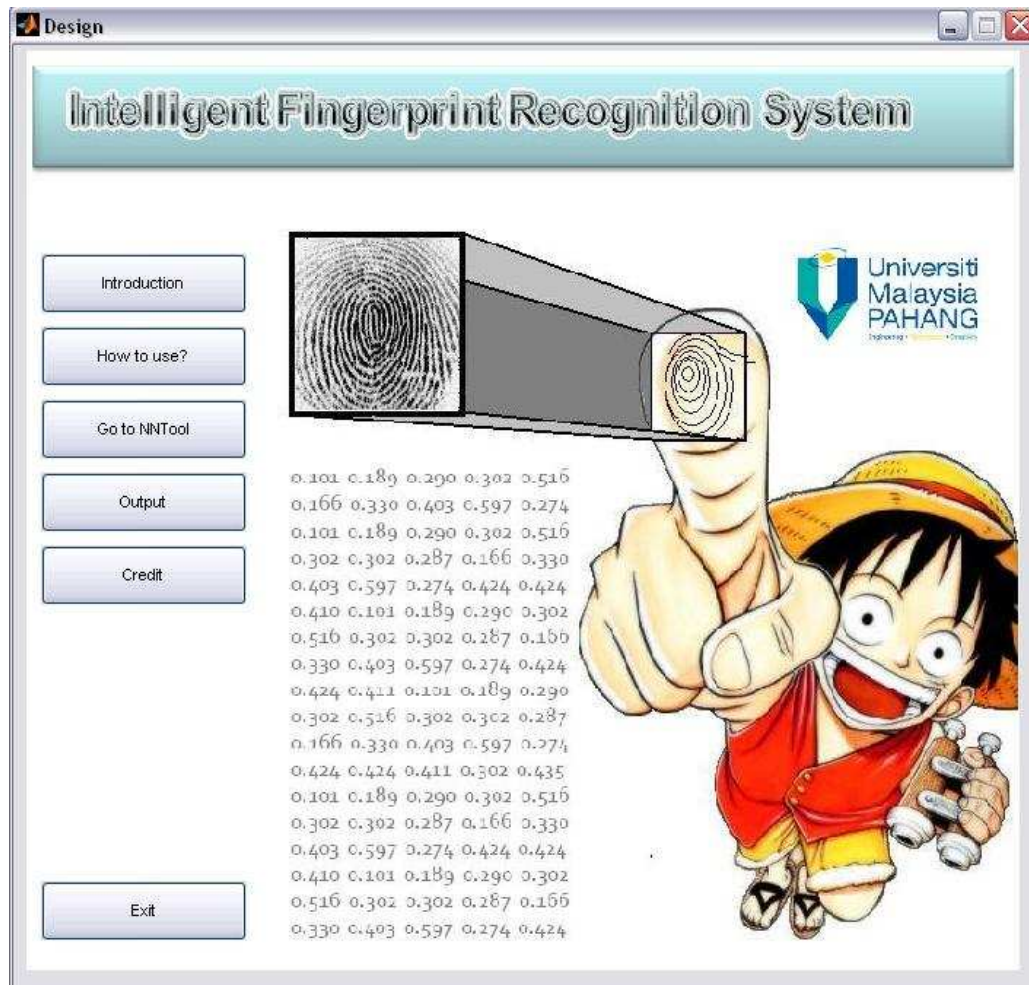


Figure 4.1 GUIDE Main Page

4.3 Recognition Results

4.3.1 Fingerprint Type A

After using and implement the Neural Network toolbox, the output will be produce and the matrix form is defined by the M-file. So, the result will produce a message box.

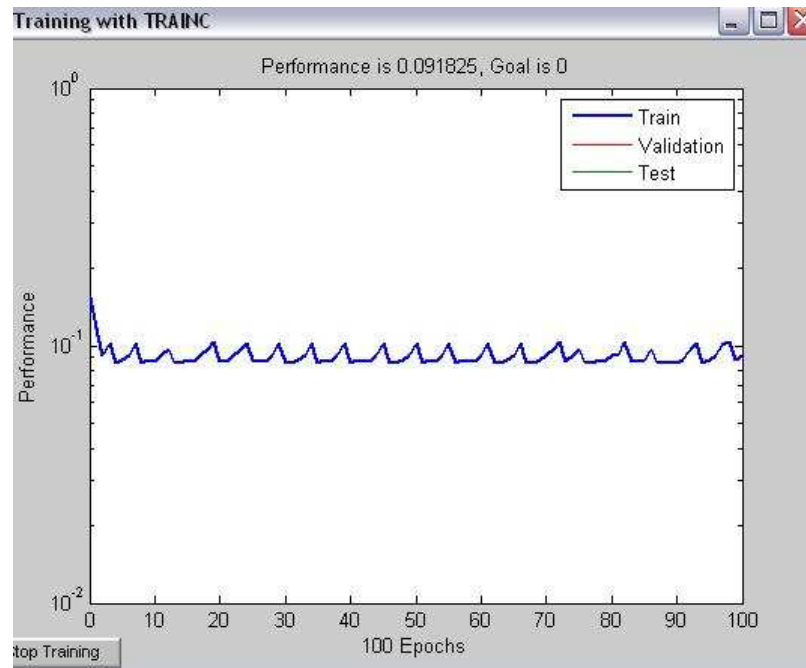


Figure 4.2 Performance of Type A

Graph above shows that the value of the performance to the data type A. The value of performance is 0.091825.

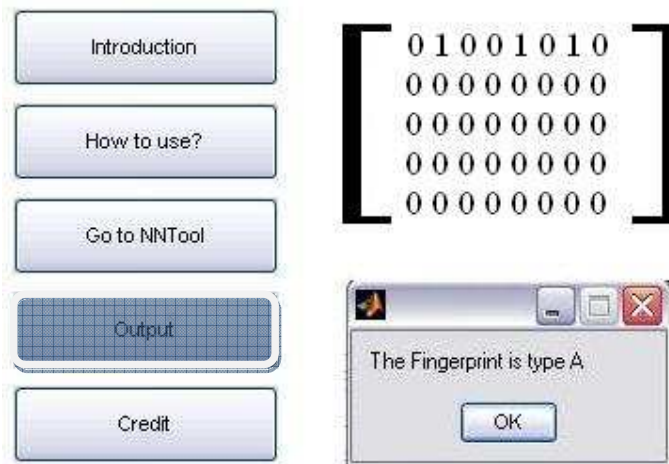


Figure 4.3 Output Display for Type A

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.3 above.

4.3.2 Fingerprint Type B

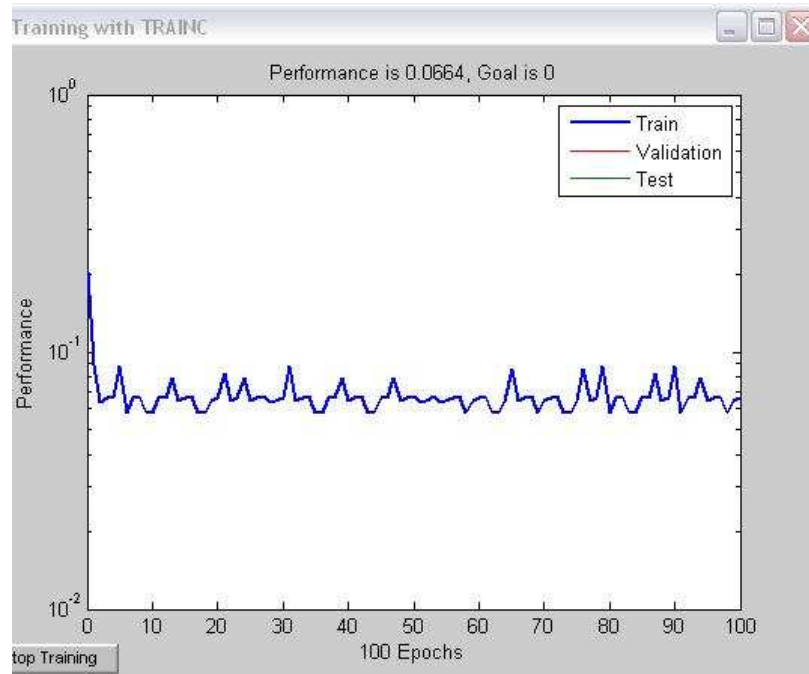


Figure 4.4 Performance of Type B

Graph above shows that the value of the performance to the data type B. The value of performance is 0.0664.

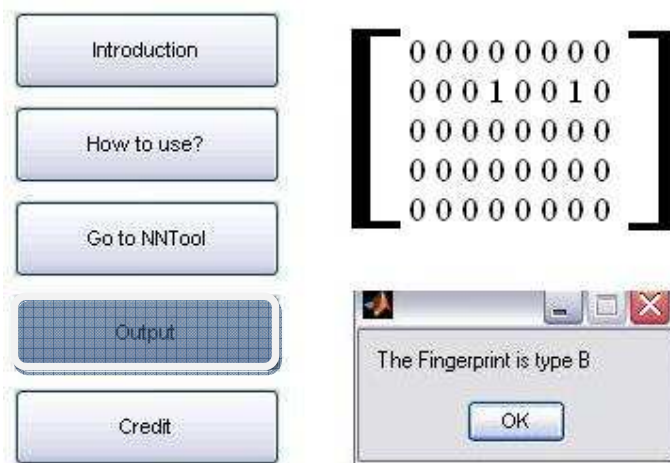


Figure 4.5 Output Display for Type B

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.5 above.

4.3.3 Fingerprint Type C

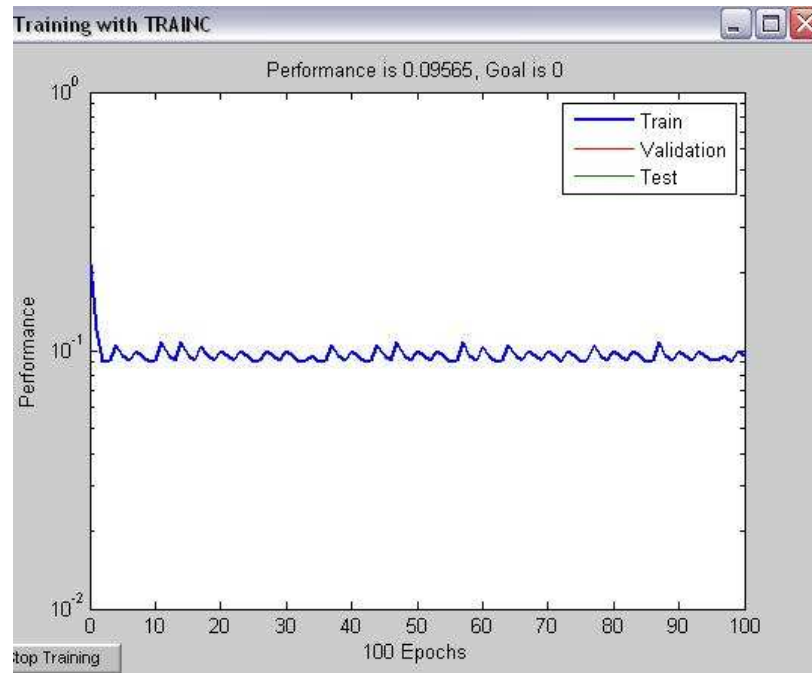


Figure 4.6 Performance of Type C

Graph above shows that the value of the performance to the data type C. The value of performance is 0.09565.

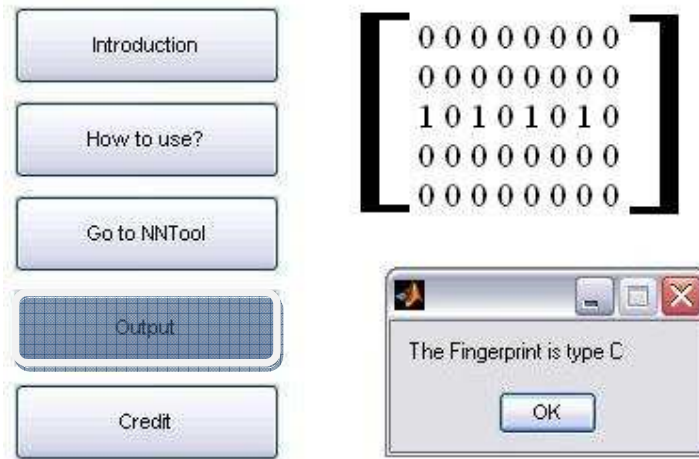


Figure 4.7 Output Display for Type C

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.7 above.

4.3.4 Fingerprint Type D

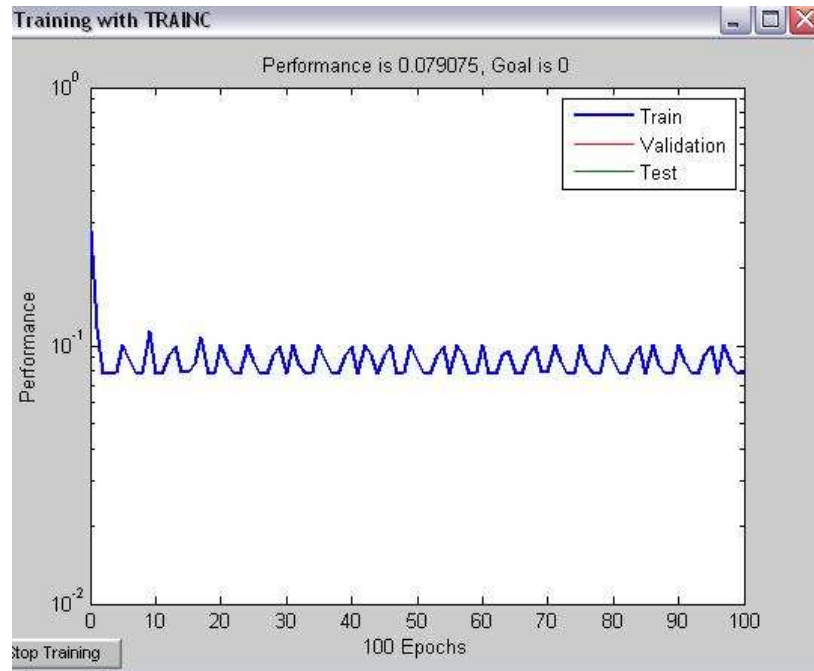


Figure 4.8 Performance of Type D

Graph above shows that the value of the performance to the data type D. The value of performance is 0.079075

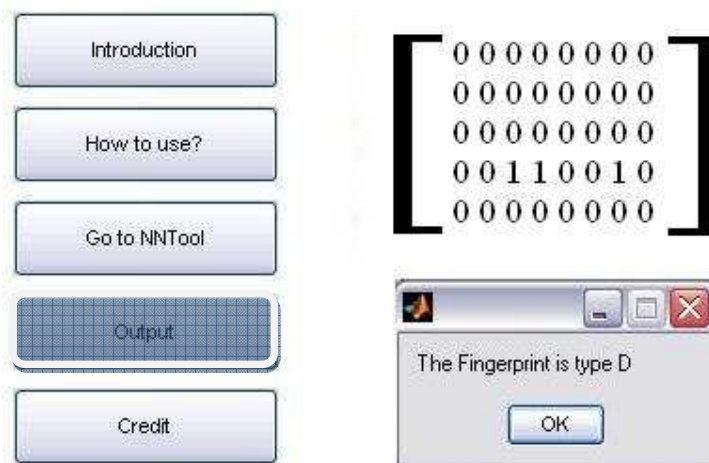


Figure 4.9 Output Display for Type D

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.9 above.

4.3.5 Fingerprint Type E

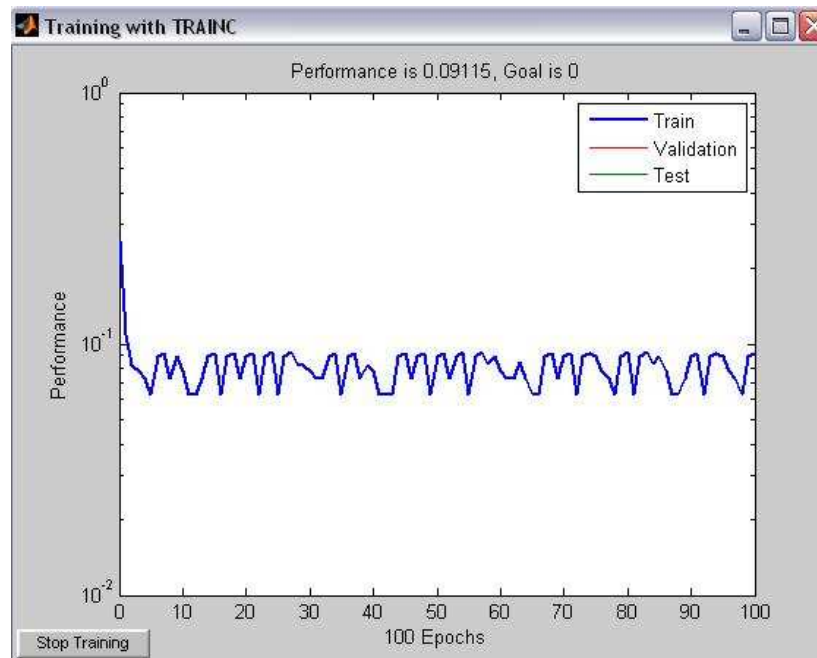


Figure 4.10 Performance of Type E

Graph above shows that the value of the performance to the data type E. The value of performance is 0.09115.

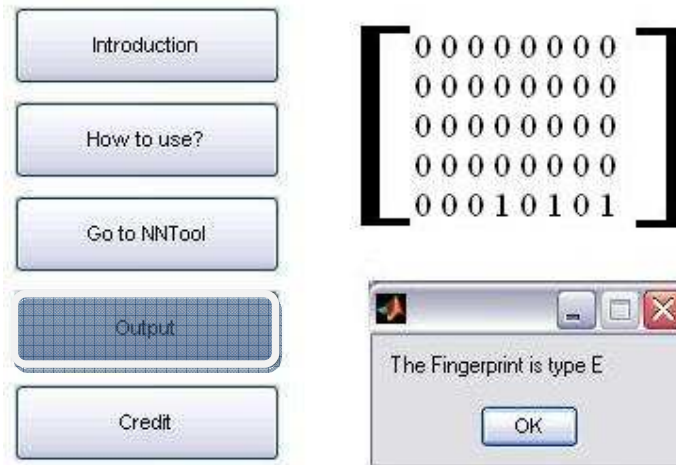


Figure 4.11 Output Display for Type E

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.11 above.

4.3.6 Unrecognized Fingerprint

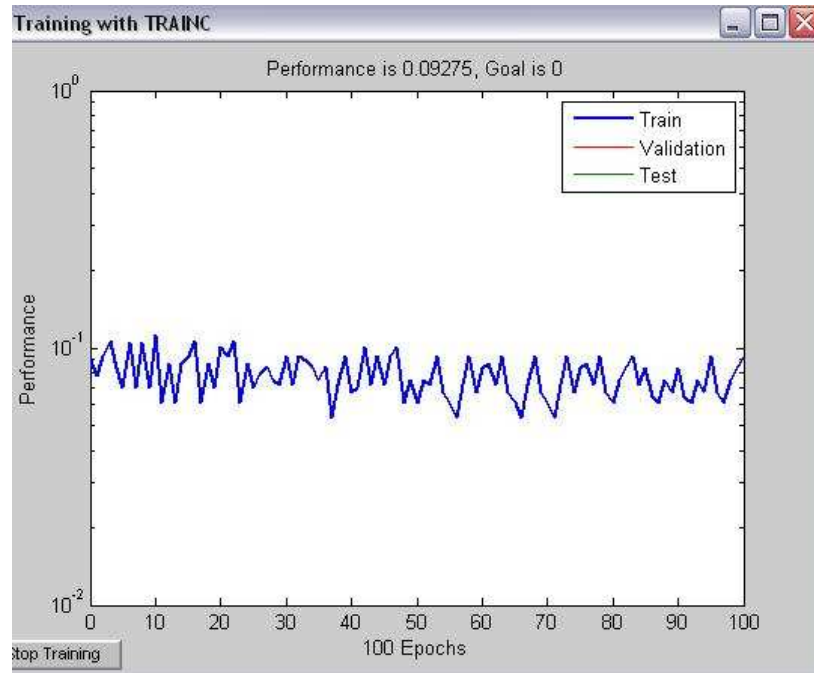


Figure 4.12 Performance of Unrecognized Type

Graph above shows that the value of the performance to the random data. The value of performance is 0.09275.

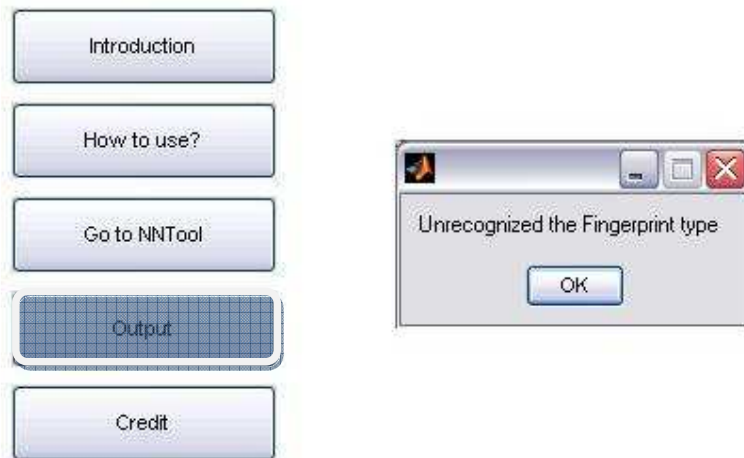


Figure 4.13 Output Display for Unrecognized Type

After export the output result from the data manager of the NNTool, refer to the main page of the GUI and click the 'Output' push button to display the result. The message box will show as Figure 4.13 above.

4.4 Close System

At the end of using the system, it needs to be close and from that the exit button is created. This exit button will close all application.



Figure 4.14 Exit Button Display

4.5 Discussion

From this project, there are many element must be consider and focus in training the data until it produce maintain and static value. The problem using this application is difficult to understand and implement. Since the GUI and M-file are constructed to state the difficulty output value in the easy statement to understand. There are many difficulties in searching the training data. This data also can be extract from the image by our own. But, it is need other system application such as data mining and image processing.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Introduction

Conclusion and recommendation are made after the analyses of the result are done. The conclusion involve of the summary of the project taken. Then, recommendations are prepared for future progress and improvement of the projects done.

5.2 Conclusion

Finally the project is successfully done with recognizing the fingerprint data by using Neural Network toolbox. Almost using the Perceptron is not a complete system of using Neural Network which is need more than one application in Neural Network toolbox such as Feedfoward, Backpropagation, Hopfield and others, the outputs were display.

Neural Network normally could be constructing by more five hundreds data. It is difficult to develop that much of data which need to train more than 10 times for each data. Perceptron is a sub-application of the main applications such as Feedfoward and Backpropagation. But user also can construct by making a Multilayer Perceptron (MLP).

The usage of GUIDE in making Graphical User Interface (GUI) is making of any system inside or outside the MATLAB software more easy to use. That is called User-friendly. In this Fingerprint Recognition System, it successful of interface the NNTool and produce the output display by import the output result from the workspace of the MATLAB software.

5.3 Recommendation

For a future development of this project, the Neural Network Toolbox can be design to recognize above 500 data and variety types of image such as face expression, footprints, cornea, ear print and others. This project is most perfectly if we combine all the applications in Neural Network Toolbox such as Feedforward, Backpropagation and others. This Neural Network toolbox must be interface with Graphical User Interface (GUI) in GUIDE. Furthermore, the output result must be display just a single click after choosing or insert the raw data.

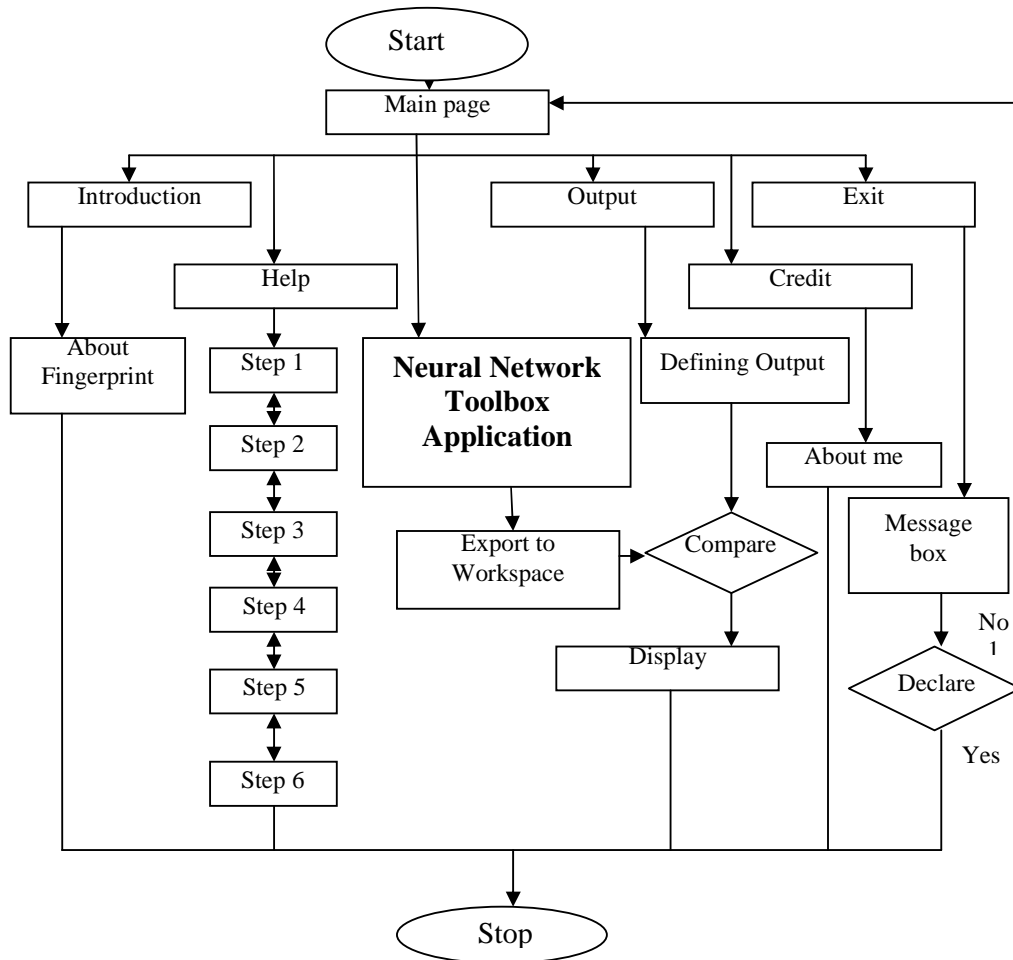
REFERENCES

1. Ashbaugh, David R. *Ridgeology. Journal of Forensic Identification Vol 41, 1995*, ISSN: 0895-173X
2. 16 May 2007, Citing Internet Source URL. *Definition of Fingerprint*.
<http://en.wikipedia.org/wiki/Fingerprint>
3. 16 May 2007, Citing Internet Source URL. *Types of Fingerprint*.
<http://biometrics.cse.msu.edu/fingerprint.html>
4. NRC (1999), "*Developments in Artificial Intelligence*", *Funding a Revolution: Government Support for Computing Research*, National Academy Press.
5. Charniak, E. and McDermott, D., *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
6. 24 March 2007, Citing Internet Source URL. *Neural Networks*.
http://www.mathtools.net/MATLAB/Neural_Networks
7. 12 April 2007, Citing Internet Source URL. *Neural Network in MATLAB*.
http://en.wikipedia.org/wiki/Neural_network_software
8. 17 March 2007, Citing Internet Source URL. *Pattern Recognition*.
http://en.wikipedia.org/wiki/Pattern_recognition
9. 1 October 2007, Citing Internet Source URL. *Graphical User Interface (GUI)*.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/buildgui.pdf
10. 3 October 2007, Citing Internet Source URL. *Advantage of GUI*.
<http://www.ncd.gov/newsroom/publications/pdf/gui.pdf>

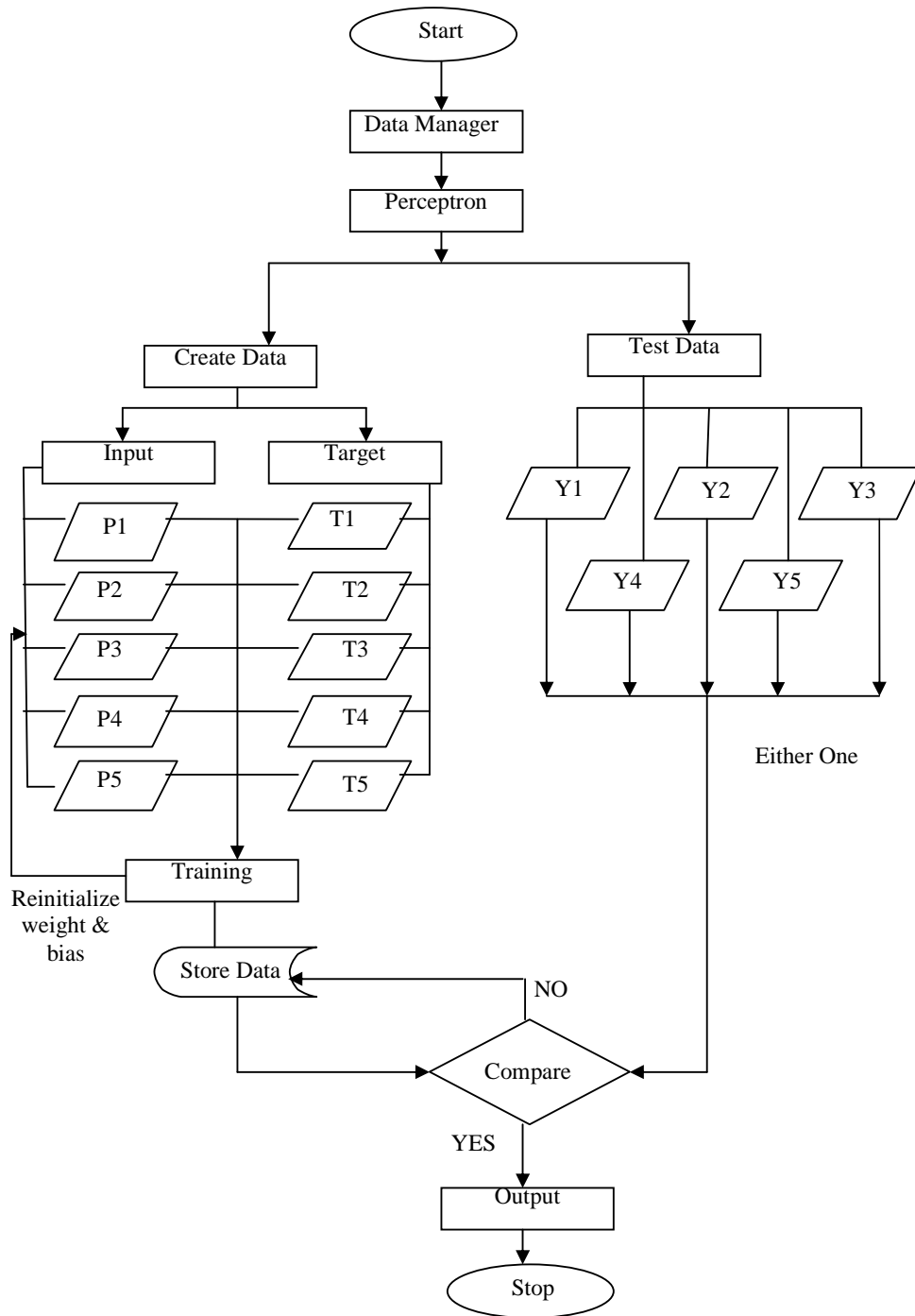
APPENDIX A

FLOWCHART OF THE GUI AND NEURAL NETWORK TOOLBOX

1) FLOWCHART OF THE GRAPHICAL USER INTERFACE (GUI) IN GUIDE



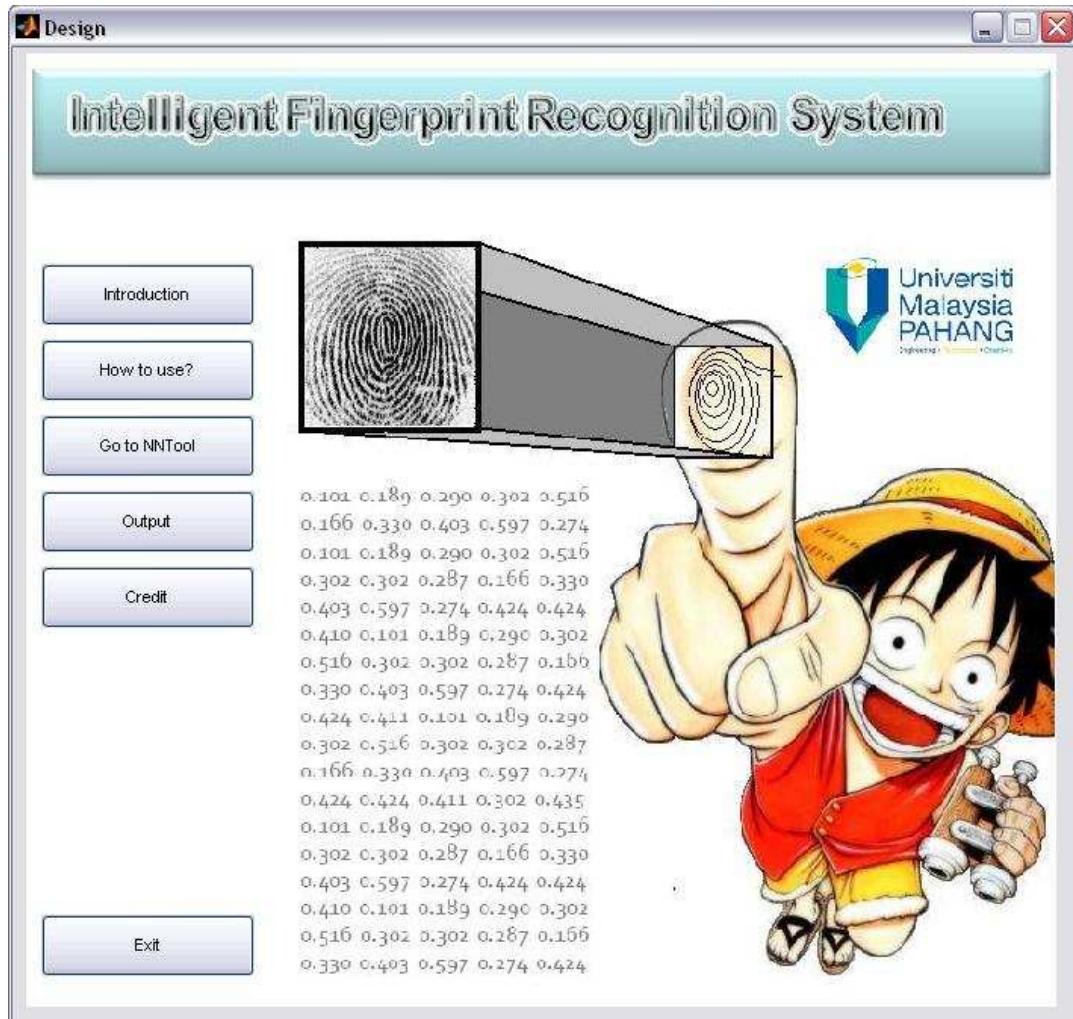
2) FLOWCHART OF NEURAL NETWORK TOOLBOX



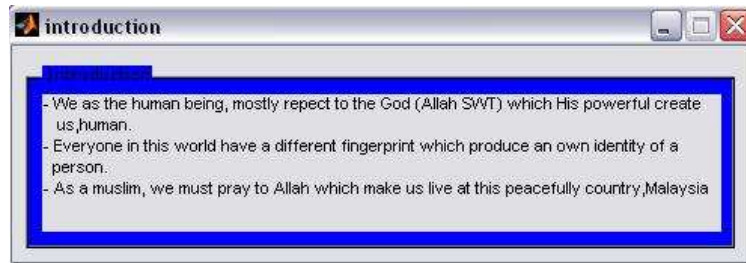
APPENDIX B

IMAGES OF GUI APPLICATION

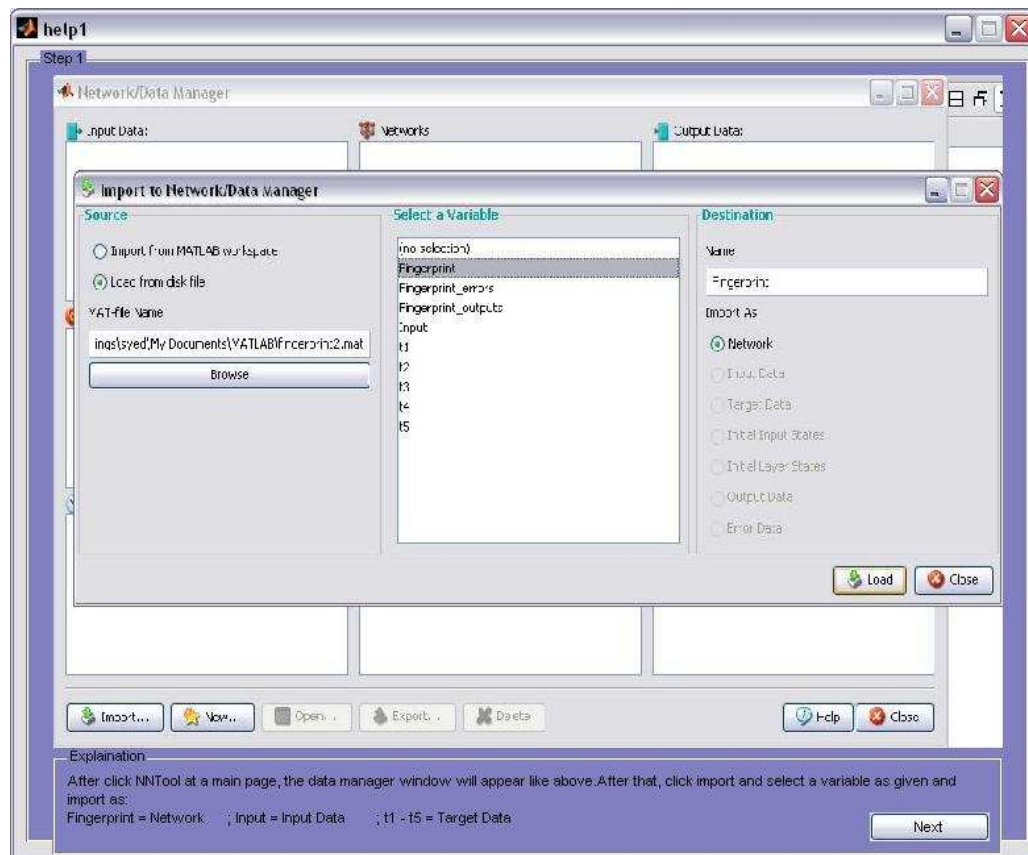
1) MAIN PAGE

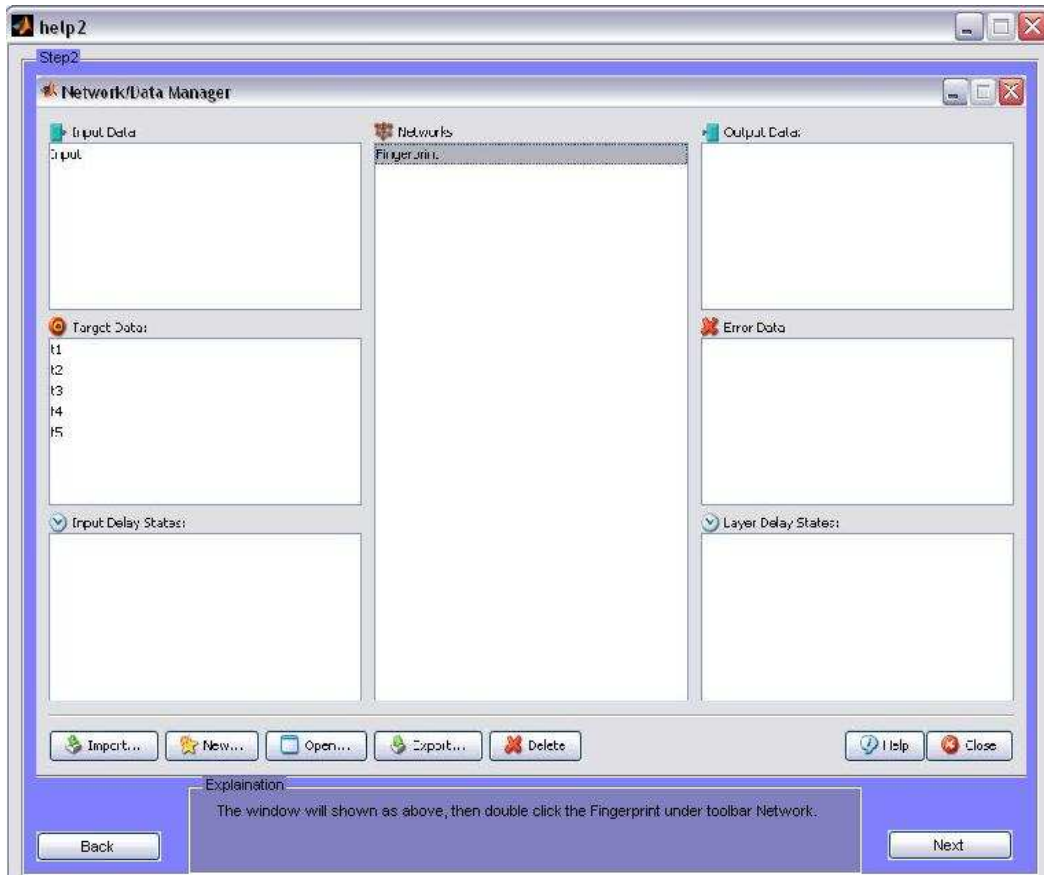


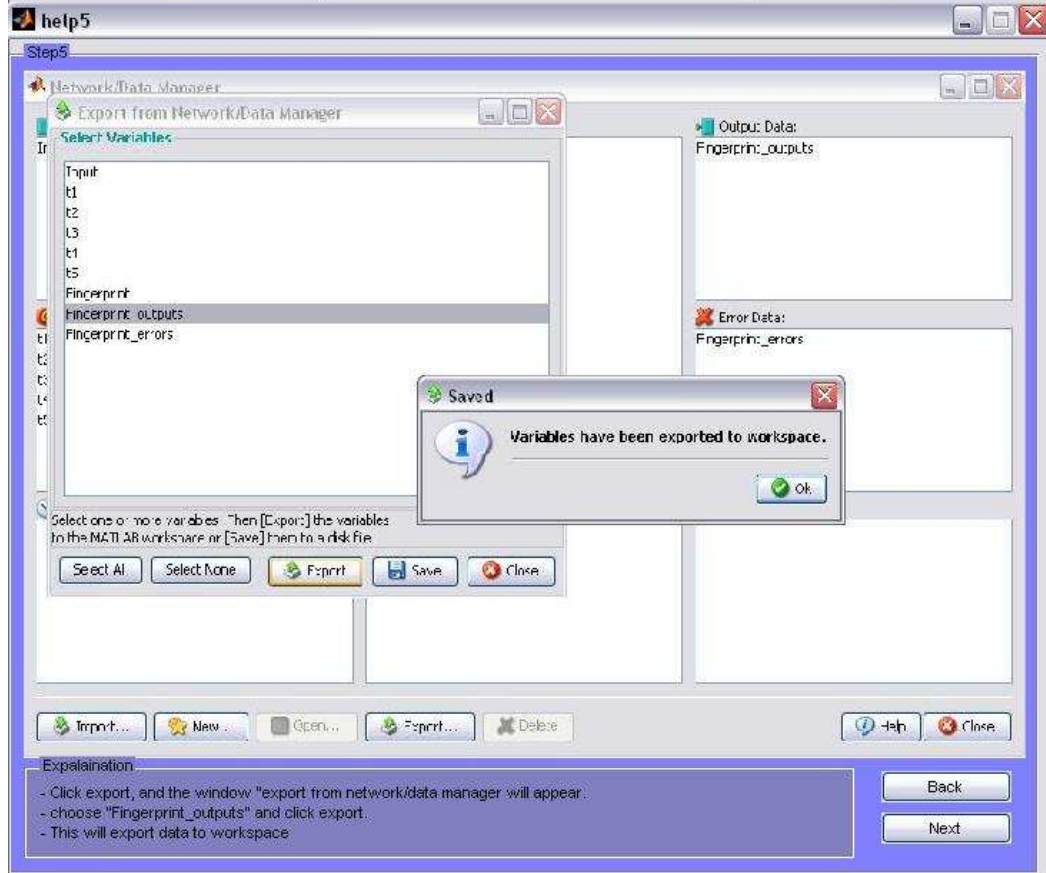
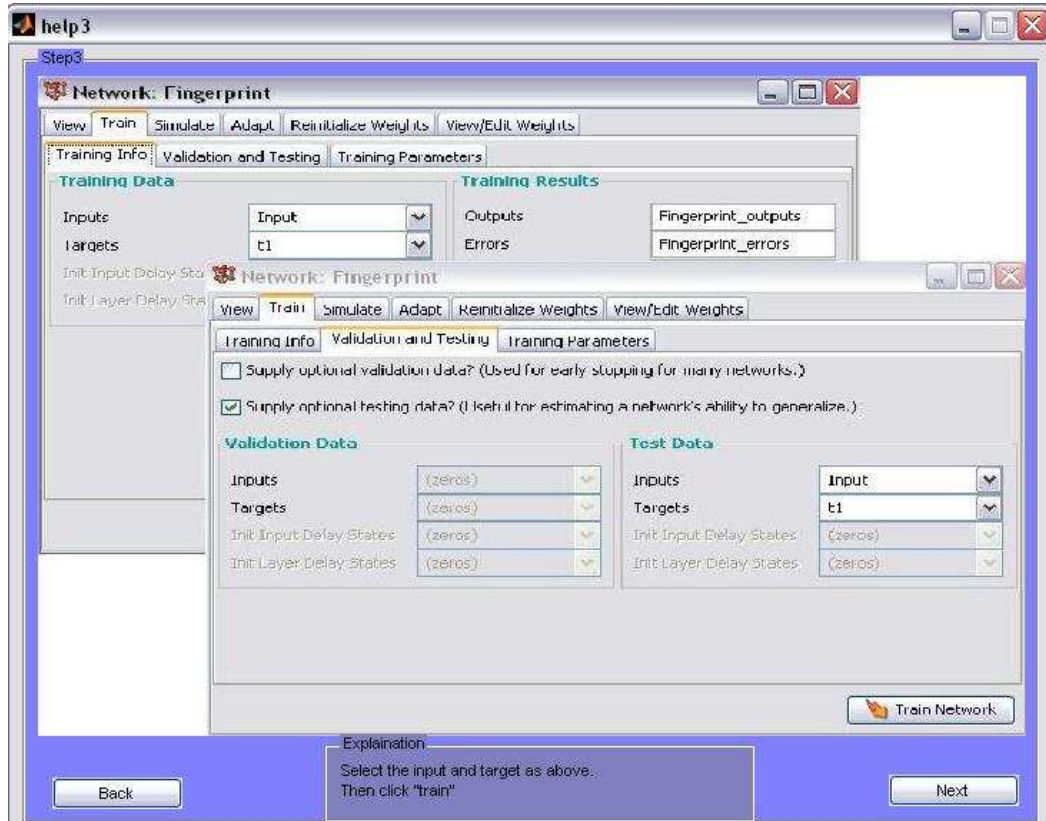
2) INTRODUCTION

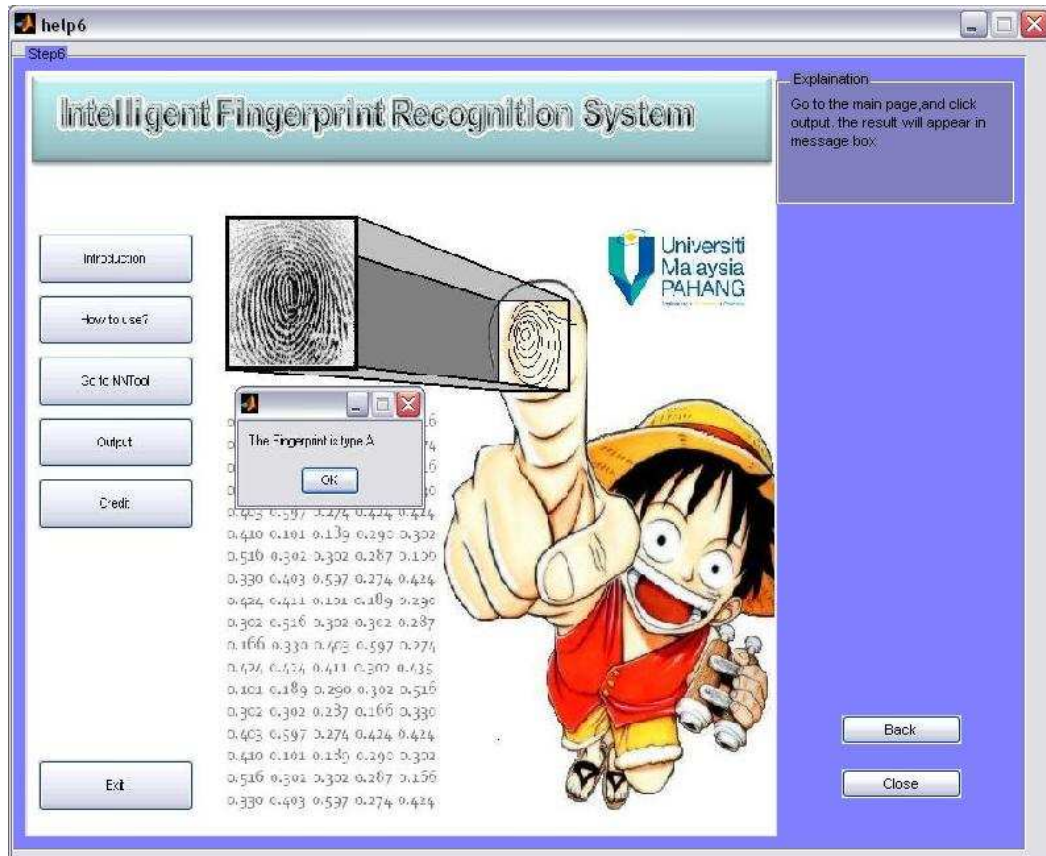


3) HELP









4) CREDIT



APPENDIX C

PROGRAMMING OF GRAPHICAL USER INTERFACE (GUI)

1) MAIN PAGE

```
function varargout = Design(varargin)
% DESIGN M-file for Design.fig
%   DESIGN, by itself, creates a new DESIGN or raises the existing
%   singleton*.
%
%   H = DESIGN returns the handle to a new DESIGN or the handle to
%   the existing singleton*.
%
%   DESIGN('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in DESIGN.M with the given input arguments.
%
%   DESIGN('Property','Value',...) creates a new DESIGN or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before Design_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to Design_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Design

% Last Modified by GUIDE v2.5 28-Oct-2007 21:28:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @Design_OpeningFcn, ...
```

```

        'gui_OutputFcn', @Design_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Design is made visible.
function Design_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Design (see VARARGIN)

% Choose default command line output for Design
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Design wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% to make a background just only use axes by resize fit to the work size
[x,map]=imread('jejari','jpg');
image(x)

```

```
set(gca,'visible','off')
```

```
% --- Outputs from this function are returned to the command line.
```

```
function varargout = Design_OutputFcn(hObject, eventdata, handles)
```

```
% varargout cell array for returning output args (see VARARGOUT);
```

```
% hObject handle to figure
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
```

```
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton2.
```

```
% link the button with the new GUI figure. just the introduction
```

```
function varargout=pushbutton2_Callback(h, eventdata, handles, varargin)
```

```
figure (introduction)
```

```
% hObject handle to pushbutton2 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton4.
```

```
% link the button with the new GUI figure. show the procedure in using
```

```
% NNtoool
```

```
function varargout=pushbutton4_Callback(h, eventdata, handles, varargin)
```

```
figure (help1)
```

```
% hObject handle to pushbutton4 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton5.
```

```
% link the button with the other GUI concept in neural network toolbox. new
```

```

% windows of the GUI will appear. otherwise it also can be display with
% manually called.
function varargout=pushbutton5_Callback(h, eventdata, handles, varargin)
NNTool
% hObject handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton6.
% link the button with the figure out the same function in window command.
function pushbutton6_Callback(hObject, eventdata, handles)
a=evalin ('base','Fingerprint_outputs')

if a>=[0 1 0 0 1 0 1 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0]
    message=strcat('The Fingerprint is type A');
    msgbox(message);

elseif a>=[0 0 0 0 0 0 0 0;0 0 0 1 0 0 1 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0]
    message=strcat('The Fingerprint is type B');
    msgbox(message);

elseif a>=[0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;1 0 1 0 1 0 1 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0]
    message=strcat('The Fingerprint is type C');
    msgbox(message);

elseif a>=[0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 1 1 0 0 1 0;0 0 0 0 0 0 0 0]
    message=strcat('The Fingerprint is type D');
    msgbox(message);

```

```

elseif a>=[0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0; 0 0 0 1 0 1 0
1]
    message=strcat('The Fingerprint is type E');
    msgbox(message);

else
    message=strcat('Unrecognized the Fingerprint type');
    msgbox(message);

end

% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton7.
% link the button with the new GUI figure. show the profile of the designer
% and his supervisor
function varargout=pushbutton7_Callback(h, eventdata, handles, varargin)
figure (credit)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton8.
% function to quit the system.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

button = questdlg(' Are you sure to exit?','Fingerprint Recognition','Yes','No','No');
switch button
    case 'Yes',

```

```
close all
case 'No',
    quit cancel;
end
clc
disp('Thank You for using this System');
```

2) INTRODUCTION

```
function varargout = introduction(varargin)
% INTRODUCTION M-file for introduction.fig
%   INTRODUCTION, by itself, creates a new INTRODUCTION or raises the
existing
%   singleton*.
%
%   H = INTRODUCTION returns the handle to a new INTRODUCTION or the
handle to
%   the existing singleton*.
%
%   INTRODUCTION('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in INTRODUCTION.M with the given input
arguments.
%
%   INTRODUCTION('Property','Value',...) creates a new INTRODUCTION or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before introduction_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to introduction_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help introduction

% Last Modified by GUIDE v2.5 28-Oct-2007 22:51:00

% Begin initialization code - DO NOT EDIT
```

```

gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @introduction_OpeningFcn, ...
                  'gui_OutputFcn', @introduction_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before introduction is made visible.
function introduction_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to introduction (see VARARGIN)

% Choose default command line output for introduction
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes introduction wait for user response (see UIRESUME)

```



```
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = introduction_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

3) HELP

```
function varargout = introduction(varargin)
% INTRODUCTION M-file for introduction.fig
%   INTRODUCTION, by itself, creates a new INTRODUCTION or raises the
existing
%   singleton*.
%
%   H = INTRODUCTION returns the handle to a new INTRODUCTION or the
handle to
%   the existing singleton*.
%
%   INTRODUCTION('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in INTRODUCTION.M with the given input
arguments.
%
%   INTRODUCTION('Property','Value',...) creates a new INTRODUCTION or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before introduction_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to introduction_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help introduction

% Last Modified by GUIDE v2.5 28-Oct-2007 22:51:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```

gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @introduction_OpeningFcn, ...
                  'gui_OutputFcn', @introduction_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before introduction is made visible.
function introduction_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to introduction (see VARARGIN)

% Choose default command line output for introduction
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes introduction wait for user response (see UIRESUME)
% uiwait(handles.figure1);

```

```
% --- Outputs from this function are returned to the command line.  
function varargout = introduction_OutputFcn(hObject, eventdata, handles)  
% varargout cell array for returning output args (see VARARGOUT);  
% hObject handle to figure  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
  
% Get default command line output from handles structure  
varargout{1} = handles.output;
```

4) CREDIT

```
function varargout = credit(varargin)
% CREDIT M-file for credit.fig
%   CREDIT, by itself, creates a new CREDIT or raises the existing
%   singleton*.
%
%   H = CREDIT returns the handle to a new CREDIT or the handle to
%   the existing singleton*.
%
%   CREDIT('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in CREDIT.M with the given input arguments.
%
%   CREDIT('Property','Value',...) creates a new CREDIT or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before credit_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to credit_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help credit

% Last Modified by GUIDE v2.5 29-Oct-2007 01:12:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @credit_OpeningFcn, ...
```

```

        'gui_OutputFcn', @credit_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before credit is made visible.
function credit_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to credit (see VARARGIN)

% Choose default command line output for credit
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes credit wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% import image to be display the pic
s = imread('aku','jpg');
axes(handles.axes1);

```

```
imshow(s);
```

```
handles.s=s;
```

```
guidata(hObject,handles);
```

```
s = imread('nieha','jpg');
```

```
axes(handles.axes2);
```

```
imshow(s);
```

```
handles.s=s;
```

```
guidata(hObject,handles);
```

```
% --- Outputs from this function are returned to the command line.
```

```
function varargout = credit_OutputFcn(hObject, eventdata, handles)
```

```
% varargout cell array for returning output args (see VARARGOUT);
```

```
% hObject handle to figure
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
```

```
varargout{1} = handles.output;
```

```
% --- Executes on slider movement.
```

```
function slider1_Callback(hObject, eventdata, handles)
```

```
% hObject handle to slider1 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'Value') returns position of slider
```

```
% get(hObject,'Min') and get(hObject,'Max') to determine range of slider
```

```
% --- Executes during object creation, after setting all properties.
```

```
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```