

# **E-Handrawn Calculator**

**By**

**Syamimi Binti Mohamad  
8479**

**Final Report submitted in partial fulfillment of  
the requirements for the  
Bachelor of Information Technology (Hons)  
(Business Information Systems)**

**JULY 2008**

**Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan**

# CERTIFICATION OF APPROVAL

E-Handrawn Calculator

by

Syamimi Binti Mohamad

A project Final Report submitted to the  
Information Technology Programme  
Universiti Teknologi PETRONAS  
in partial fulfillment of the requirement for the  
BACHELOR OF TECHNOLOGY (Hons)  
(BUSINESS INFORMATION SYSTEM)

Approved by,



(Mr. Jale B. Ahmad)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2008

## **CERTIFICATION OF ORIGINALITY**

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

**SYAMIMI BINTI MOHAMAD**

## ABSTRACT

This draft for final report is about my research progress in developing e-Hand-Drawn Calculator as final year project. The purpose of this project is to demonstrate an application of back-propagation network (comparison of training their algorithms and transfer function) in order to developing e-Hand-Drawn Calculator. Back-propagation network is a supervised learning method, and is an implementation of the Delta rule. It requires a teacher that knows, or can calculate, the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Backpropagation requires that the activation function used by the artificial neurons (or "nodes") is differentiable. The main activities in this project are Assemble the training data, Create the network object, Train the network and Simulate the network response to new inputs. The training data consist of ten sample of each number zeros until number nine and symbol plus, minus, division and multiplication. These all data will be train, testing and validation before enter to next stage which is creating network object. This section presents the architecture of the network that is most commonly used with the backpropagation algorithm; the multilayer feedforward network. The investigation for combination of Neuron Model (tansig, logsig, purelin) and training algorithms (traingd, traingdm, traingda, traingdx, trainrp, traingcp, traingcb, traingcg, traingbf, traingoss, trainglm, traingbr); tend to know which combination will give the greatest result and smallest error.

## ACKNOWLEDGEMENT

First and foremost, all my gratitude to The Almighty, Allah SWT that had given me the strength, wisdom and patient all the way in completing this project within the time given.

My heartfelt thanks goes to everyone who has been supportive and giving me their point of views in realizing the project. Without the existence of many people around, the project would reach to nothing.

Special thanks to my supervisor, Ms. Jale B. Ahmad who has been understanding, supportive, patient and helping me a lot from the beginning to the end.

Thanks also to all my friends that had been very supportive and helpful during the development process of the application. Thanks for the ideas, critics, comment and suggestion to improve this project especially my colleagues that have shared the same experience in making each individual's Final Year Project a success.

And last but not least, million thanks to my parent who has been very understanding and always give me full support and encouragement especially when I almost giving up.

Thank you to everybody involves along the development process of the project. Thanks for sharing all the valuable experience, knowledge and ideas. Thank you.

## **Table of Contents**

<b>CERTIFICATE OF APPROVAL</b>	<b>ii</b>
<b>CERTIFICATION OF ORIGINALITY</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>ACKNOWLEDGEMENT</b>	<b>v</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>CHAPTER 1: INTRODUCTION</b>	
1.1 Background of Study	1
1.2 Problem Statement	3
1.3 Objectives and Scope of Study	6
1.3.1 Objectives	6
1.3.2 Scope of Study	7
1.4 Relevancy of Study	8
<b>CHAPTER 2: LITERATURE REVIEW</b>	
2.0 Related Research	9
2.1 Neural Network	11
2.1.1 The Multi Layer Perceptron	12
2.2 Feed-Forward back-Propagation network	13
2.3 Preprocessing	14
2.4 Recognizing a Hand-Drawn Character	15
2.5 Isolated and Continuous Digit	15
2.6 Electronic Paper	16
2.7 Digital Paper	16

### **CHAPTER 3: METHODOLOGY**

3.1 Project Phases	17
3.2 The flow of e-hand-drawn calculator system	22
3.3 Tools and Equipments	23
3.2.1 System Requirements	23
3.2.2 Software and Tools for Development	23
3.4 System Development Schedule	24

### **CHAPTER 4: RESULT AND DISCUSSION**

4.1 Bank of Images	25
4.2 Training	27
4.3 Testing Data	30
4.4 Validation Data	32
4.5 Creating Network Data	34
4.6 Simulation Data	39
4.8 Result	40
4.9 Setting up of GUI	43
4.10 Idea of E-Handrawn's GUI Functionality	44

### **CHAPTER 5: CONCLUSION AND RECOMMENDATIONS**

5.1 Conclusion	45
5.2 Recommendations	46

### **CHAPTER 6: REFERENCES** 47

### **CHAPTER 7: APPENDICES**

7.1 Appendix A :GanttChart	49
7.2 Result of Training Algorithm	50

## List of Figure

Figure 1: Show the neural networks are adjusted, or trained, so that particular input leads to a specific target output.	11
Figure 2: Multi Layer Perceptron	12
Figure 3: isolated digit	16
Figure 4: mis-segmentation Digit	16
Figure 5 : discontinuous digit	16
Figure 6 : Continuous digit	16
Figure 7: the flow of e-hand-drawn calculator system	23
Figure 9: Ten Samples of number zero “0” and “7”	26
Figure10: Image Number Attribute	27
Figure11 : Figure 11: Graph of Training Algorithm Result	42
Figure 12: Graphical User Interface	43
Figure 13: Picture or input before system resize	44
Figure 14: Picture or input after system resized	44
Figure 15: Milestone for the First Semester of 2 -Semester Final Year Project	49

## List of Table

Table 1: List of System Requirements	23
Table 2: List of Software and Tools for Development	23

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Study

The main concern of the study is to develop an interactive calculator by using neural network fundamental and implementation in MATLAB. That calculator is called The e-Hand-Drawn Calculator, it is a hand-writing calculator and it applies in computer. For initiate, this calculator just has a few function such as plus (+), minus (-), multiply (x) and division ( $\div$ ) and e-Hand-Drawn Calculator's function will be add in the future to increase its professionalism. The principle of develop this kind of hand-drawn calculator because to achieve human natural technique or natural way of calculation rather than click the mouse or press the keyboard

This e-Hand-Drawn Calculator is developing by using backpropagation technique for feed forward Neural Network. These are input, hidden, and output Layers. During the training phase, the training data is fed into to the input layer. The data is propagated to the hidden layer and then to the output layer. This is called the *forward pass* of the back propagation algorithm. In *forward pass*, each node in hidden layer gets input from all the nodes from input layer, which are multiplied with appropriate weights and then summed. The output of the hidden node is the nonlinear transformation of the resulting sum. Similarly each node in output layer gets input from all the nodes from hidden layer, which are multiplied with appropriate weights and then summed. The output of this node is the non-linear transformation of the resulting sum.

The output values of the output layer are compared with the *target output values*. The *target output values* are those that we attempt to teach our network. The error between actual output values and target output values is calculated and propagated back toward hidden layer. This is called the *backward pass* of the back propagation algorithm. The error is used to update the connection strengths between nodes, i.e. weight matrices between input-hidden layers and hidden-output layers are updated.

During the testing phase, no learning takes place i.e., weight matrices are not changed. Each test vector is fed into the input layer. The feed forward of the testing data is similar to the feed forward of the training data.

## **1.2 Problem Statement**

- **Existing calculator in computer not approach to natural way of human calculation**

Existing calculator in computer is use click mouse technique and presses the keyboard; it quiet difficult for human because in nature, human do the calculation with hand-writing. That approved by mathematics history, since mathematics was discovered by paleontologists in ancient Egyptian and Babylonian epoch; Egyptian and Babylonian start do the mathematics with write down the numbers and symbol on caves' wall and rock surface.

Of course, writing faster rather than click number and symbol using mouse because or press the keyboard because user need to find where location of numbers and symbols; that situation will waste time and little bit irritating for user.

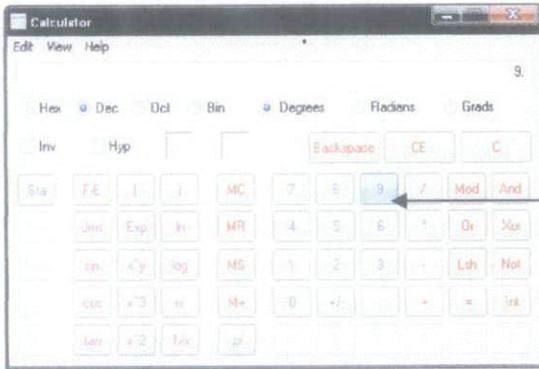
- **Existing calculator in computer not user-friendly**

Furthermore, existing calculator in computer cannot display the all equation that users have composed, from this situation; users will be confusing and they maybe make a second count for same mathematic equation.

For example:

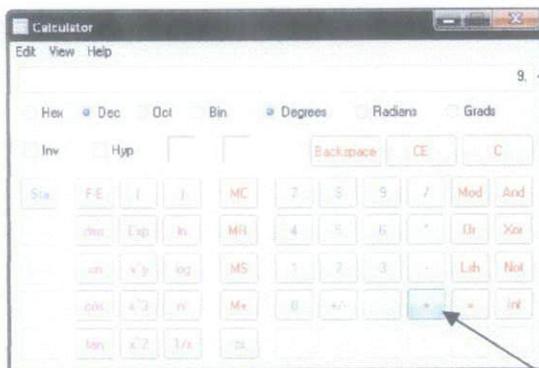
Below is a sample of calculation of  $9 + 1 = 10$  by using calculator in a computer.

1)



User enters number 9

2)

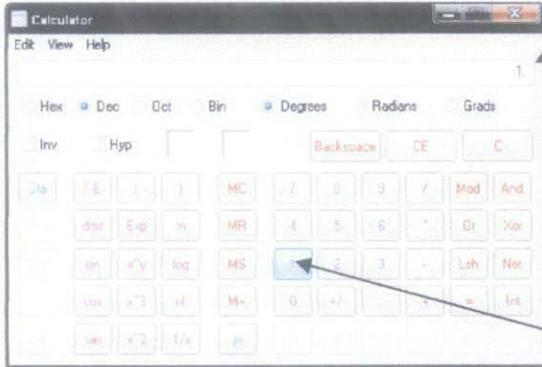


2) But calculator not display the "+" sign

Will make user confuse what kind of sign that has inserted

1) User enters "+" signs for addition function

3)

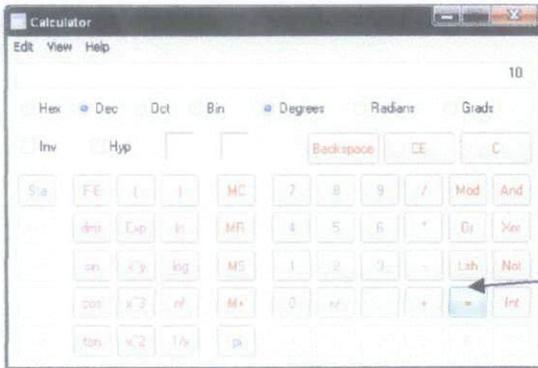


2) The previous number 9 was disappeared and replace with number 1.

Will make user confuse what previous number that has inserted

1) Then user enters number 1

4)



Calculator show the answer 10 after user click "=" sign.

## **1.3 Objectives and Scope of Study**

### **1.3.1 Objectives**

The objectives of this project are:

- To develop a hand-writing calculator as alternative calculator in computer application.

Existing calculator in computer is using the clicking mouse technique and presses the keyboard; it quiet difficult for human because in nature, human do the calculation with hand-writing.

- To develop a calculator that approach human natural way of calculation.

Since naturally human do the calculating by using their hand then e-handrawn calculator is the suitable system that can fulfill the human desire and need which is this system can functioning as a calculator by using human hand writing.

- Later on this project can be integrated in electronic paper (e-paper) or Digital Paper since nowadays its application is inadequate.

In view of the fact that calculator function is does not apply yet in electronic paper (e-paper) or Digital Paper so the e-handrawn calculator is very appropriate to implement into electronic paper (e-paper) or Digital Paper since they using same technique which is hand writing recognition.

### 1.3.2 Scope of Study

Focus on studying the neural network fundamental in MATLAB to developing e-handrawn calculator. These include:

- ✦ Creating 10 sample of each numbers 0-9 and 10 sample of each symbol of +, -, ÷, x by using paint and all of them have same size which are 100 x 100 pixels. Also they all have saved as bitmap images.
- ✦ Developing MATLAB coding as e-hand-drawn calculator system able to read all folders of numbers and character that have been drawn.
- ✦ Converting number.bmp into matrix style by using backpropagation fundamental.
- ✦ Testing the e-hand-drawn calculator system with variation of input
- ✦ Creating coding for this system be able functioning as a calculator
- ✦ The design of the interface that is accessible for everyone

The system is very complex in terms of developing coding with backpropagation algorithm and system must perform as a calculator.

## 1.4 Relevancy of Study

The study is match to Information System (IS) field because it is concentrates with system development. The study will start with understanding the concept of developing a system, analysis the existing research and system which related to pattern recognition by using fundamental of neural network in MATLAB . Then the study continues with the analyzing requirement and designing the solution. Finally, the study is about doing the programming and coding. The study allows IS student to improve their programming skills in many different languages and allows students to apply the real situation in developing software.

MATLAB is a numerical computing environment and programming language. Created by The Math Works, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it specializes in numerical computing, an optional toolbox interfaces with the Maple symbolic engine, allowing it to be part of a full computer algebra system.

This e-hand-drawn calculator use MATLAB application which is use a lot of MATLAB coding in store the images (character and symbol), train the system to recognize the images (character and symbol) and testing the system whether the system accomplish accordingly.

## CHAPTER 2

### LITERATURE REVIEW

Review for the study was taken abundantly from journals, internet, books and opinion & teaching from my supervisor; **Mr. Jale Ahmad**. The main focus is on neural network fundamental especially backpropagation network because this system using this application to functioning respectively.

#### 2.0 Related Research

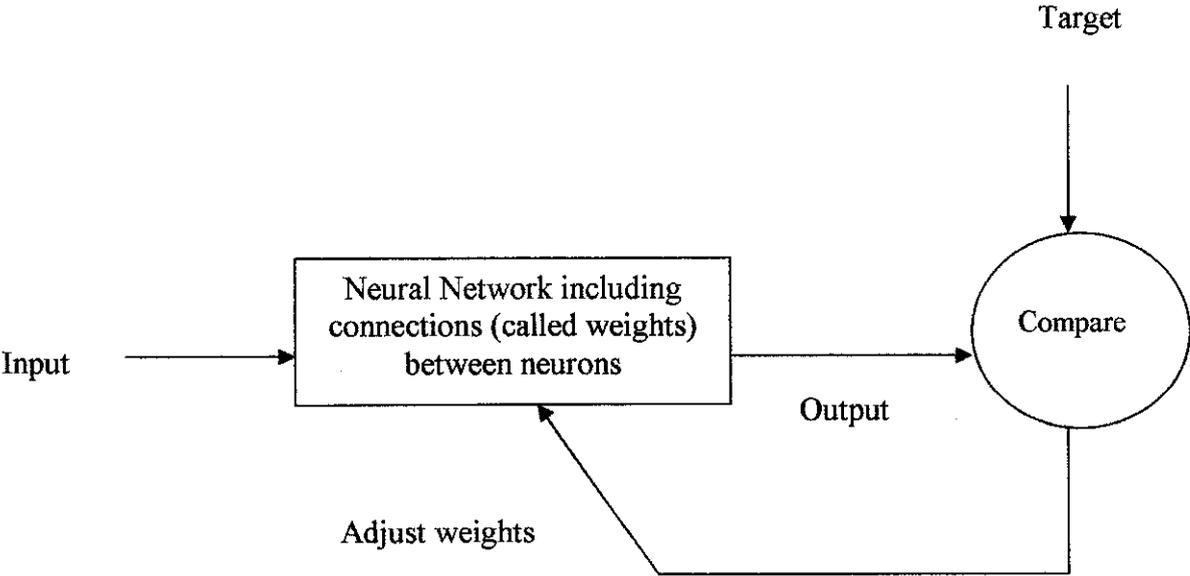
The existing developed system that used fundamental of neural network and yet, this system was found in Malaysian Journal of Computer Science, Vol. 17 No. 2, December 2004, pp. 40-54. The title of this system is **DIGIT RECOGNITION USING NEURAL NETWORKS** by Chin Luh Tan and Adznan Jantan from Faculty of Engineering Universiti Putra Malaysia, 43400 Serdang, Selangor Darul Ehsan; Malaysia. This paper investigates the use of feed-forward multi-layer perceptrons trained by back-propagation in speech recognition. Besides this, the paper also proposes an automatic technique for both training and recognition. The use of neural networks for speaker independent isolated word recognition on small vocabularies is studied and an automated system from the training stage to the recognition stage without the need of manual cropping for speech signals is developed to evaluate the performance of the automatic speech recognition (ASR) system. Linear predictive coding (LPC) has been applied to represent speech signal in frames in early stage. Features from the selected frames are used to train multilayer perceptrons (MLP) using back-propagation.

The same routine is applied to the speech signal during the recognition stage and unknown test patterns are classified to the nearest patterns. In short, the selected frames represent the local features of the speech signal and all of them contribute to the global similarity for the whole speech signal. The analysis, design and development of the automation system are done in MATLAB, in which an isolated word speaker independent digits recognizer is developed.

**2.1 Neural Network**

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

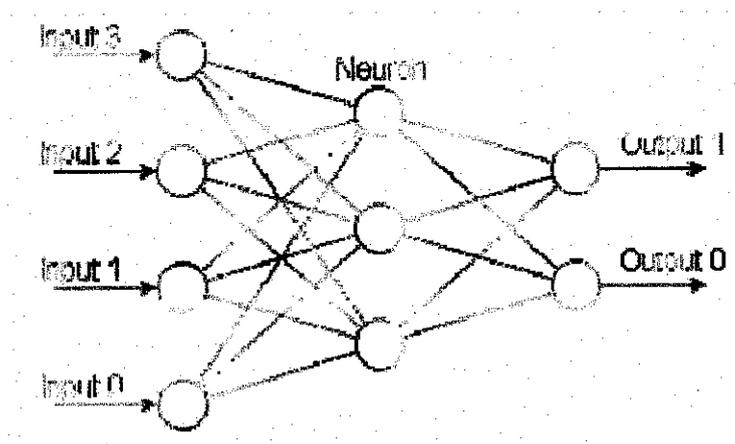
Commonly, neural networks are adjusted, or trained, so that particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this supervised learning, to train a network.



*Figure 1: Show the neural networks are adjusted, or trained, so that particular input leads to a specific target output.*

### 2.1.1 The Multi Layer Perceptron

Multi-layer perceptrons are one of many different types of existing neural networks. They comprise a number of neurons connected together to form a network. The “strengths” or “weights” of the links between the neurons is where the functionality of the network resides. Its basic structure is shown in *Figure 2*.



*Figure 2: Multi Layer Perceptron*

The idea behind neural networks stems from studies of the structure and function of the human brain. Neural networks are useful to model the behaviors of real-world phenomena. Being able to model the behaviors of certain phenomena, a neural network is able subsequently to classify the different aspects of those behaviors, recognize what is going on at the moment, diagnose whether this is correct or faulty, predict what it will do next, and if necessary respond to what it will do next.

## 2.2 Feed-Forward Back-Propagation Network

Backpropagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by you. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

Standard backpropagation is a gradient descent algorithm, as is the Widrow-Hoff learning rule, in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods. The Neural Network Toolbox implements a number of these variations. This chapter explains how to use each of these routines and discusses the advantages and disadvantages of each.

Properly trained backpropagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input/target pairs and get good results without training the network on all possible input/output pairs. There are two features of the Neural Network Toolbox that are designed to improve network generalization - regularization and early stopping.

The primary objective of this chapter is to explain how to use the backpropagation training functions in the toolbox to train feedforward neural networks to solve specific problems. There are generally four steps in the training process:

- i. Assemble the training data
- ii. Create the network object
- iii. Train the network
- iv. Simulate the network response to new inputs

### **2.3 Preprocessing**

The segmentation (separating each digit from its neighbors) would be a relatively simple task if we could assume that a character is contiguous and is disconnected from its neighbors, but neither of these assumptions holds in practice. Many ambiguous characters in the database are the result of mis-segmentation.

At this point, the size of a digit varies but typically around 100 by 100 pixels. Since the input of a back-propagation network is fixed size, it is necessary to normalize the size of characters. This performed using a linear transformation to make the characters fit in a 100 by 100 pixel images. This transformation preserves the aspect ratio of the character, and is performed after extraneous marks in the image have been removed. Because of the linear transformation, the resulting image is not binary but has multiple gray levels, since a variable number of pixels in the original image can fall into a given pixel in the target image. The grey level of each image are scaled and translated to fall within the range -1 to 1.

## 2.4 Recognizing a Hand-Drawn Character

Hand-drawn alphanumeric character recognition occurs within the boundaries of a box or boxes that you define for the input panel of your target device. Because each box represents a character or glyph, the structure of the applicable language determines how boxes are arranged. For example, the boxes could be arranged from left to right and top to bottom for English. By moving a stylus in a predefined pattern within the box, the handwriting recognition engine processes and recognizes this input, one character, or glyph at a time, and produces the corresponding Unicode output.

## 2.5 Isolated and Continuous Digit

The e-handrawn calculator system only can read continuous and isolated digit. The isolated digit means segmentation of each digit (separating each digit from its neighbors), we could assume that a character is contiguous and is disconnected from its neighbors. The continuous digit here means the number input must unbroken digit. Below is the example of isolated digit and continuous digit:

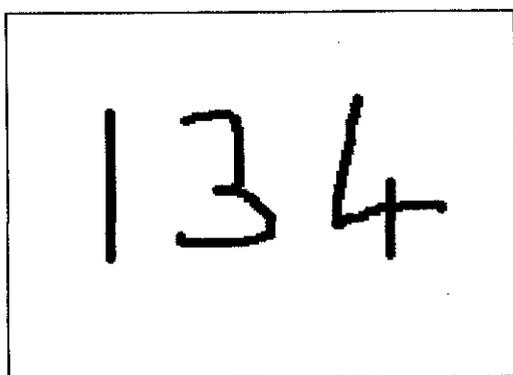


Figure 3: isolated digit

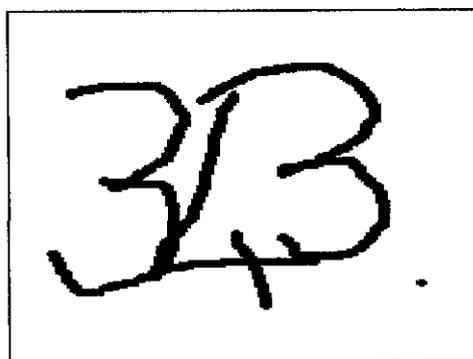


Figure 4: mis-segmentation digit.

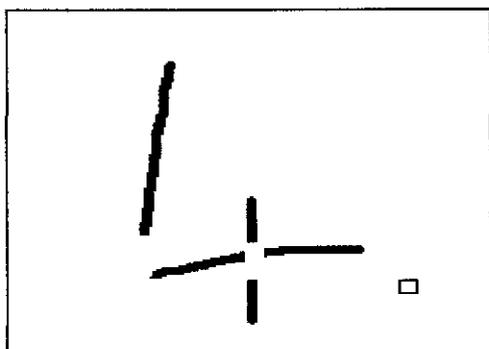


Figure 5 : discontinuous digit

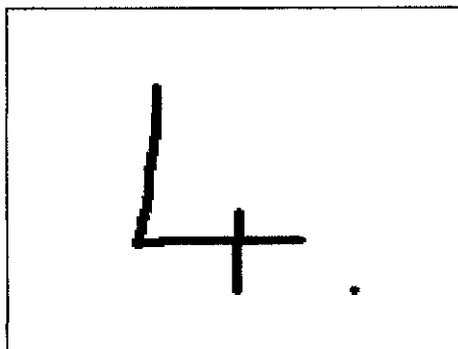


Figure 6 : Continuous digit

## 2.6 Electronic Paper

Electronic paper, also called e-paper, is a display technology designed to mimic the appearance of ordinary ink on paper. Unlike a conventional flat panel display, which uses a backlight to illuminate its pixels, electronic paper reflects light like ordinary paper and is capable of holding text and images indefinitely without drawing electricity, while allowing the image to be changed later.

## 2.7 Digital Paper

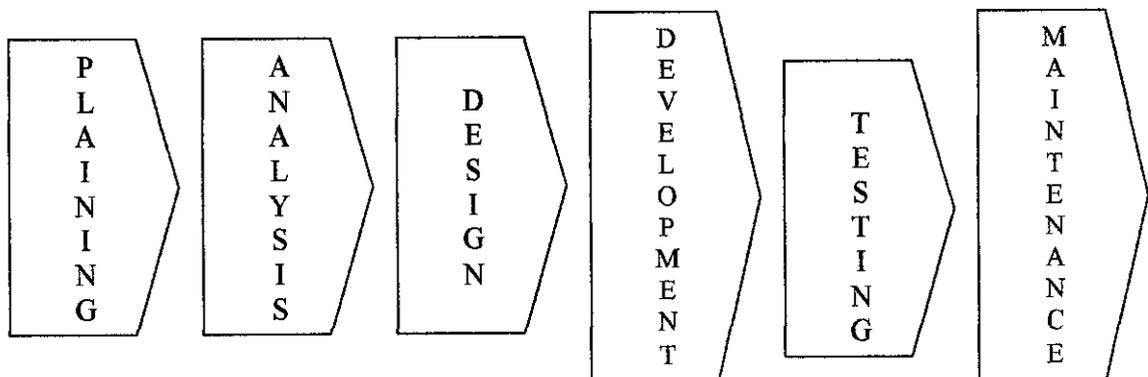
Digital paper, also known as interactive paper, is patterned paper used in conjunction with a digital pen to create handwritten digital documents. The printed dot pattern uniquely identifies the position coordinates on the paper. The digital pen uses this pattern to store the handwriting and upload it to a computer.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Project Phase

The methodology of this project is using System Development life Cycle (SDLC), the detail methodology as below:



## *PLANNING*

The main tasks of this project are:

- Creating 10 sample of each numbers 0-9 and 10 sample of each symbol of +, -, ÷, x by using paint and all of them have same size which are 100 x 100 pixels. Also they all have saved as bitmap images.
- Developing MATLAB coding as e-hand-drawn calculator system able to read all folders of numbers and character that have been drawn.
- Converting number.bmp into binary code by using backpropagation fundamental.
- Do the training data.
- Testing the e-hand-drawn calculator system with variation of input
- Do data validation to To make confirmation of system performance to recognize the all database (image number)
- Creating coding for this system be able functioning as a calculator
- Creating Network object and simulation
- The design of the interface that is accessible for everyone

## *ANALYSIS*

Before go further in developing system, other important part is analysis the project. A few things that we should analyze are:

- The e-hand-drawn calculator project ought to fulfill the Final Year Project Requirement such as no plagiarism and must be original.
- Skill that developer of this system necessity for example able to use MATLAB application.
- Study the training algorithm, transfer function and network hidden layer in order to get higher performance of the e-handrawn calculator system.

## *DESIGN*

Design in methodology is means where the technical blueprint of the system is create by:

- Designing of technical architecture – the software that will be use to develop e-hand-drawn calculator is MATLAB and the hardware that system require is any type of digital mouse pen which can use for hand-writing.
- Design e-handrawn calculator system coding.
- Design the systems model – graphical user interface (GUI) for e-hand-drawn calculator.

## DEVELOPMENT

The main of development tasks of this project are:

- draw & store image (ten sample of number 0 until 9 and symbol +,-,X ,=, ÷)
- Train the system to recognize that storing created images with Neural Network concept in MATLAB.
- Testing the system with variety of input.
- Do data validation to To make confirmation of system performance to recognize the all database (image number)
- Creating coding for this system be able functioning as a calculator
- Creating Network object and simulation

- The design of the interface that is accessible for everyone
- System should perform mathematic function which initiate, the system able to do plus minus operation, division and multiplication.

## ***TESTING***

Testing the developed system:

- Creating a set of new bank image to test the system either system successfully can recognize the image number respectively.
- Test the e-hand-drawn calculator system using the establish test scripts- test conditions are conducted by comparing expected outcomes to actual outcomes. If these differ, a bug is generated and backtrack to the development stage must occur.

## ***MAINTENANCE***

Keeping the system up to date and ensuring it meets the Final Year Project goal.

**3.2 Below are the steps and explanation on the development of e-handrawn calculator system**

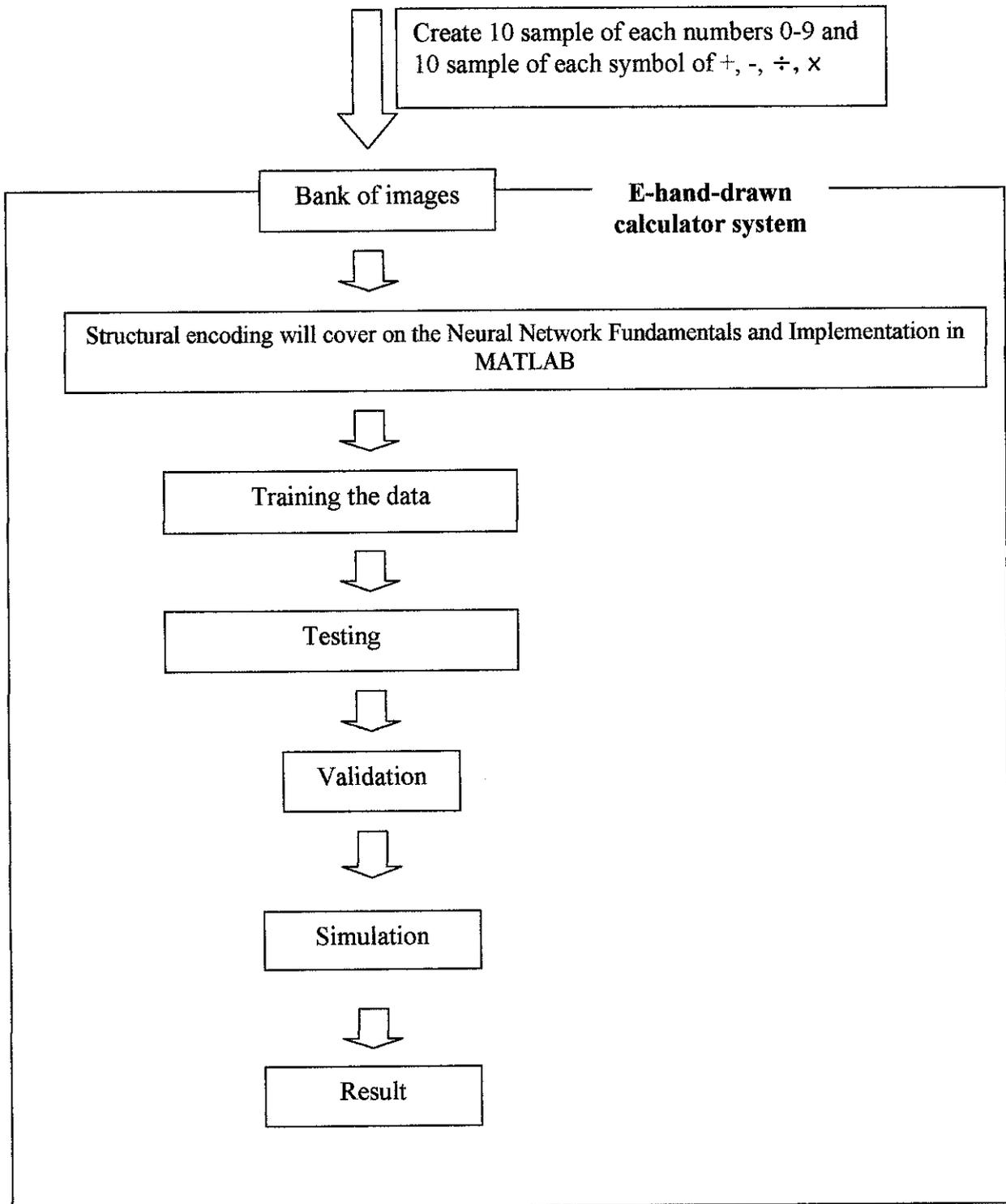


Figure 7: the flow of e-hand-drawn calculator system

### 3.3 Tools and Equipments

#### 3.3.1 System Requirements

Device/Software/Tools	Requirements
Operating System	Windows XP Pro Service Pack 2
Processor	Intel Celeron M processor 370, 1.80 GHz, 400 MHz, 1MB L2 cache
Disk Space	40 GB
Memory	448 MB of RAM
Peripherals	Mouse, Keyboard, Printer
Network	802.11 b/g wireless LAN
Web Server	IIS

Table 1: List of System Requirements

#### 3.3.2 Software and Tools for Development

Software/Tools	Description
MATLAB	Neural network fundamental, backpropagation, training and testing the system, creating interface.
Paint	Creating 10 sample of each numbers 0-9 and 10 sample of each symbol of +, -, ÷, x

Table 2: List of Software and Tools for Development

### **3.4 System Development Schedule**

The system was assigned to be completely finished within one year or two semesters time period. For the first semester, the requirements need to be defined, system needs to be designed and first prototype would be developed. In second semester, it is all about prototype refinement until the prototype become as a complete system for final presentation. See Appendix A for complete Gantt chart.

# CHAPTER 4

## RESULT AND DISCUSSION

### 4.1 Bank of Images

For this project, I have create 10 sample of each numbers 0-9 and 10 sample of each symbol of +, -, ÷, x to teach the e-hand-drawn calculator system recognize the numbers and mathematical symbol and character. This technique has been used because different people have different type of hand writing. In artificial intelligent system, we train or teach system to recognize the character but the system can recognize more than that. The images has been drawn by using Paint and the size for each images are 100 x 100 pixels; the setting color for all images are crone or black & white and the images have save as bitmap type.

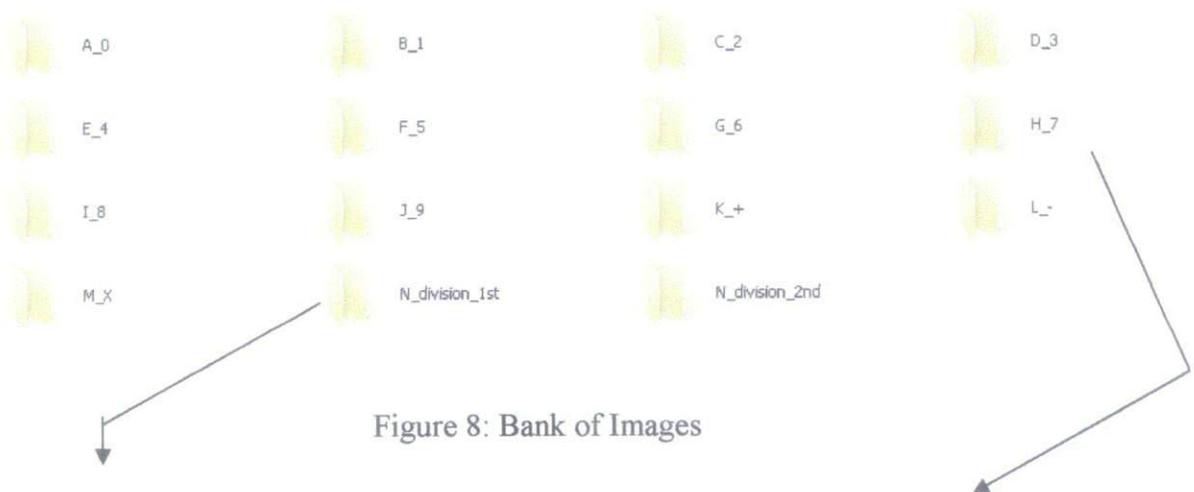


Figure 8: Bank of Images



Figure 9: Ten Samples of number zero “0” and “7”

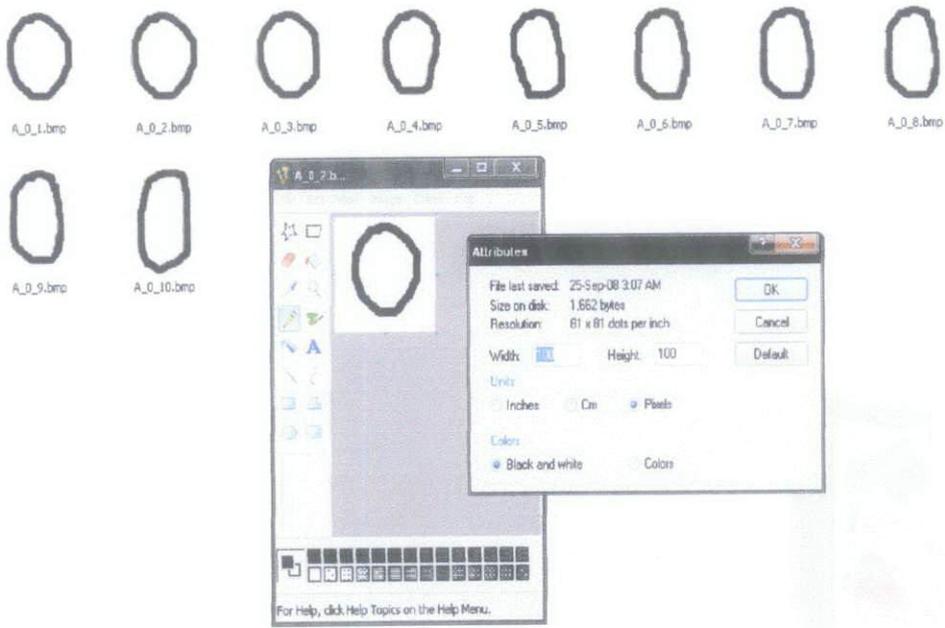
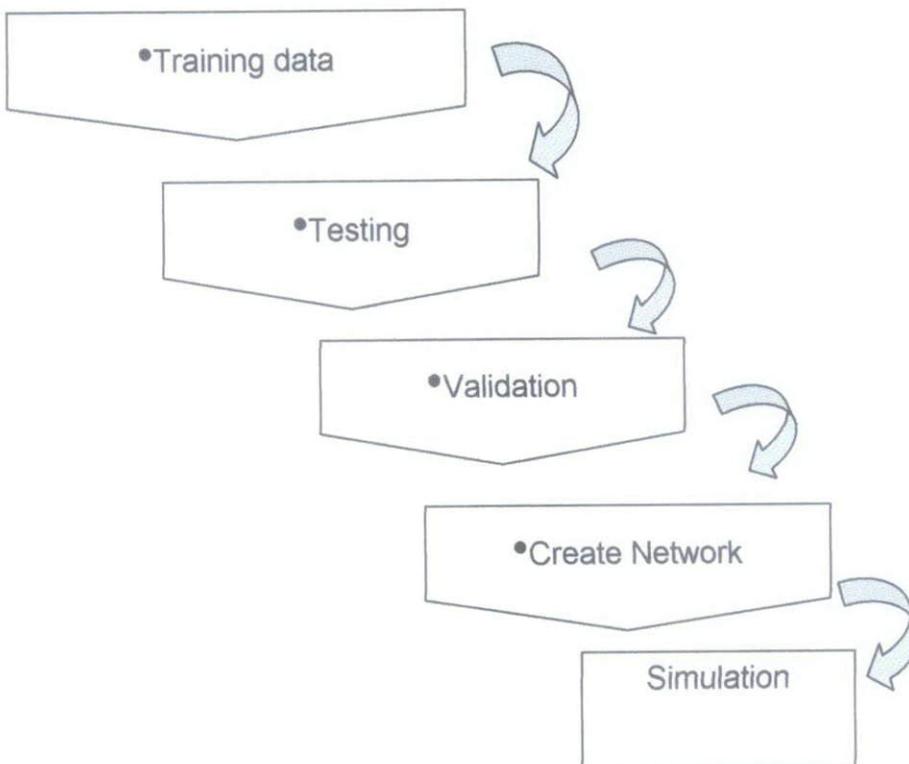


Figure10: Image Number Attribute

#### 4.11 Main Activities



## 4.2 Training Data

Standard back-propagation is a gradient descent algorithm, in which the network weights are moved along the negative of the gradient of the performance function. The term back-propagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods.

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

Main purpose of training data:

- System can read all folder number
- Converting 100 x 100 pixel of image number.bmp to binary code (0 and 1)
- Reshape
- Teach system to recognize the image number

## 4.2.1 Working Coding Training Data

```
pth='D:\fyp2_latest10\myfyp2\my fyp1 project\numbers'
myDirectory=dir(pth)
% next task : to clean . and ..
for i=1:size(myDirectory,1)-2
    myDirectory(i).name=myDirectory(i+2).name
end
bilfolder=size(myDirectory,1)-2

fprintf('Path name : %s\n', pth);
['No of Directory : ' num2str(size(myDirectory,1))]
cnt=0
maxphoto=10 % bil max photo dlm setiap folder
maxfolder=15 % bil max folder
M=[];
T=[];

for i=1:bilfolder
    if ~strcmp(myDirectory(i).name, 'Thumbs.db')

%         t=zeros(maxfolder,1); % target
        t=zeros(bilfolder,1);
        t(i,1)=1;

        fprintf('Nama directory %s \n', myDirectory(i).name)
        myphoto=dir([pth '\' myDirectory(i).name]); %nama dir
%         cnt=(i-1)*10
        %for j=1:size(myphoto,1)-1
        cntr=0;
        for j=1:size(myphoto,1)

            if ~strcmp(myphoto(j).name, 'Thumbs.db')
```

```

if ~myphoto(j).isdir
    fprintf('%s\n', [pth '\' myDirectory(i).name '\' myphoto(j).name])
    X=imread([pth '\' myDirectory(i).name '\' myphoto(j).name]);
    cnt=cnt+1;
    %figure, imshow(X);
    %cnt=cnt+1;
    %subplot(10,15,cnt), imshow(X);

    X=reshape(X,size(X,1)*size(X,2),1);%column vector
    X=imcomplement(X);
    M=[M X];
    T=[T t];

    end%if
end%if
end %for j
fprintf('Bill Files : %d\n',cnt)
end %if
end %for i

save trainData.mat M T

```

### 4.3 Testing Data

Creating a set of new bank image to test the system either system successfully can recognize the image number respectively.

#### 4.3.1 Working Coding Testing Data

```
pth='D:\fyp2_latest10\myfyp2\my fyp1 project\numbers'
myDirectory=dir(pth)
% next task : to clean . and ..
for i=1:size(myDirectory,1)-2
    myDirectory(i).name=myDirectory(i+2).name
end
bilfolder=size(myDirectory,1)-2

fprintf('Path name : %s\n', pth);
['No of Directory : ' num2str(size(myDirectory,1))]
cnt=0
maxphoto=10 % bil max photo dlm setiap folder
maxfolder=15 % bil max folder
M=[];
T=[];

for i=1:bilfolder
    if ~strcmp(myDirectory(i).name, 'Thumbs.db')

%         t=zeros(maxfolder,1); % target
        t=zeros(bilfolder,1);
        t(i,1)=1;

        fprintf('Nama directory %s \n', myDirectory(i).name)
        myphoto=dir([pth '\' myDirectory(i).name]);%nama dir
%         cnt=(i-1)*10
        %for j=1:size(myphoto,1)-1
        cntr=0;
        for j=1:size(myphoto,1)

            if ~strcmp(myphoto(j).name, 'Thumbs.db')
```

```

if ~myphoto(j).isdir
    fprintf('%s\n', [pth '\\' myDirectory(i).name '\\' myphoto(j).name])
    X=imread([pth '\\' myDirectory(i).name '\\' myphoto(j).name]);

    %imshow(E:\myfyp2\my fyp1 project\numbers2);
    %figure, imshow(X);
    cnt=cnt+1;
    subplot(15,10,cnt), imshow(X);

    X=reshape(X,size(X,1)*size(X,2),1);%column vector
    X=imcomplement(X);
    Mts=[Mts X];
    Tts=[Tts c];

        end
    end
end %for

end %if
end %tutk for

save testdata.mat Mts Tts

```

## 4.4 Validation Data

To make confirmation of system performance to recognize the all database (image number)

### 4.4.1 Working Coding of Validation Data

```
pth='D:\fyp2_latest10\myfyp2\my fyp1 project\numbers'
myDirectory=dir(pth)
% next task : to clean . and ..
for i=1:size(myDirectory,1)-2
    myDirectory(i).name=myDirectory(i+2).name
end
bilfolder=size(myDirectory,1)-2

fprintf('Path name : %s\n', pth);
['No of Directory :' num2str(size(myDirectory,1))]
cnt=0
maxphoto=10 % bil max photo dlm setiap folder
maxfolder=15 % bil max folder
M=[];
T=[];

for i=1:bilfolder
    if ~strcmp(myDirectory(i).name,'Thumbs.db')

%         t=zeros(maxfolder,1); % target
        t=zeros(bilfolder,1);
        t(i,1)=1;

        fprintf('Nama directory %s \n', myDirectory(i).name)
        myphoto=dir([pth '\' myDirectory(i).name]);%nama dir
%         cnt=(i-1)*10
        %for j=1:size(myphoto,1)-1
        cntr=0;
        for j=1:size(myphoto,1)

            if ~strcmp(myphoto(j).name,'Thumbs.db')
```

```

if ~myphoto(j).isdir
    fprintf('%s\n', [pth '\' myDirectory(i).name '\' myphoto(j).name])
    X=imread([pth '\' myDirectory(i).name '\' myphoto(j).name]);

    %imshow(E:\myfyp2\my fyp1 project\numbers2);
    %figure, imshow(X);
    cnt=cnt+1;
    subplot(15,10,cnt), imshow(X);

    X=reshape(X,size(X,1)*size(X,2),1);%column vector
    X=incomplement(X);
    Mval=[Mval X];
    Tval=[Tval t];

        end
    end
end %for

    end %if
end %tick for

save valData.mat Mval Tval

```

## 4.5 Creating Network Object

The first step in training a feedforward network is to create the network object. The function `newff` creates a feedforward network. It requires four inputs and returns the network object. The first input is an  $R$  by 2 matrix of minimum and maximum values for each of the  $R$  elements of the input vector. The second input is an array containing the sizes of each layer. The third input is a cell array containing the names of the transfer functions to be used in each layer. The final input contains the name of the training function to be used.

For example, the following command creates a two-layer network. There is one input vector with two elements. The values for the first element of the input vector range between -1 and 2, the values of the second element of the input vector range between 0 and 5. There are three neurons in the first layer and one neuron in the second (output) layer. The transfer function in the first layer is tan-sigmoid, and the output layer transfer function is linear. The training function is `traingd`.

```
net=newff([-1 2; 0 5],[3,1],{'tansig','purelin'},'traingd');
```

This command creates the network object and also initializes the weights and biases of the network; therefore the network is ready for training. There are times when you may want to reinitialize the weights, or to perform a custom initialization. The next section explains the details of the initialization process.

## 4.5.1 Working Coding of Network Object

```
load brainData.mat  
load valData.mat  
load testData.mat
```

```
val.P=data10\val;  
val.T=Tval;
```

```
test.P = data10\mts;  
test.T = Tts;
```

```
neth=newf(minmax(data10M),[15 15],{'tansig','purelin'},'traingdm');
```

```
'tansig' - tansig
```

```
'traingdm' - traingdm
```

```
% size [4 15] to [10 15] 1* [4] [20 15] etc...
```

```
neth.trainParam.show = 1;
```

```
neth.trainParam.epochs = 1000;
```

```
neth = init(neth);
```

```
[neth, tr] = train(neth,data10M,T,[],[],val,test); % to try without
```

```
save nym.mat neth -append
```

Neuron Hidden  
layer

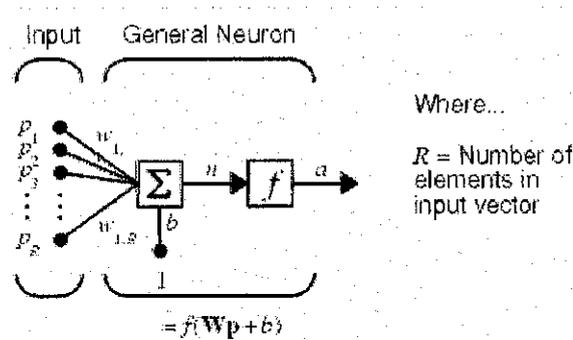
Transfer  
function

Training  
algorithm

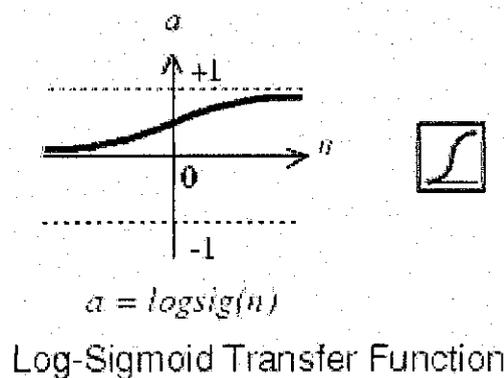
Save as

### 4.5.2 Neuron Model (tansig, logsig, purelin)

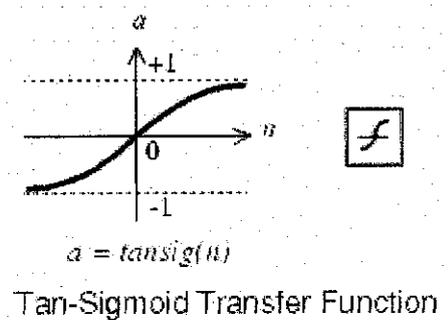
An elementary neuron with  $R$  inputs is shown below. Each input is weighted with an appropriate  $w$ . The sum of the weighted inputs and the bias forms the input to the transfer function  $f$ . Neurons may use any differentiable transfer function  $f$  to generate their output.



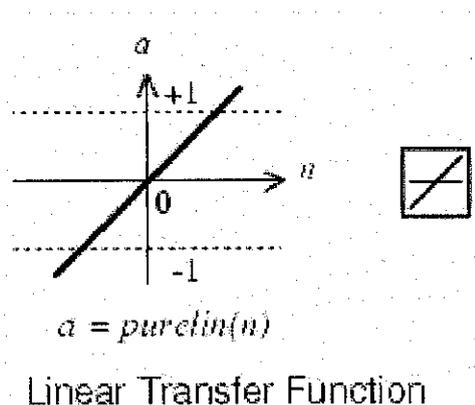
Multilayer networks often use the log-sigmoid transfer function logsig.



The function logsig generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity. Alternatively, multilayer networks may use the tan-sigmoid transfer function tansig.



Occasionally, the linear transfer function `purelin` is used in backpropagation networks.



If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value.

In backpropagation it is important to be able to calculate the derivatives of any transfer functions used. Each of the transfer functions above, `tansig`, `logsig`, and `purelin`, have a corresponding derivative function: `dtansig`, `dlogsig`, and `dpurelin`. To get the name of a transfer function's associated derivative function, call the transfer function with the string `'deriv'`, `tansig('deriv')`, `ans = dtansig`. The three transfer functions described here are the most commonly used transfer functions for backpropagation, but other differentiable transfer functions can be created and used with backpropagation if desired.

### 4.5.3 Training Algorithm

Training Algorithms		Comments
traingd	Gradient Descent(GD)	Original but slowest
Trainngdm	GD with momentum	Faster than traingd
trainгда	GD with adaptive $\alpha$	Faster than traingd, but can use for batch mode only
trainгдаx	GD with adaptive $\alpha$ and with momentum	
trainrp	Resilient backpropagation	Fast convergence and minimal storage requirements.
traincgf	Fletcher-Reeves Update	Conjugate Gradient Algorithms With fast convergence
traincgp	Polak-Ribière Update	
traincgb	Powell-Beale Restarts	
trainscg	Scaled conjugate gradient	
trainbfg	BFGS algorithm	Quasi-Newton Algorithms
trainoss	One step secant algorithm	With fast convergence.
trainlm	Levenberg-Marquardt	Fastest training. Memory reduction features
trainbr	Bayesian regularization	Improve generalization capability

## 4.6 Simulation Data

The function `sim` simulates a network. `sim` takes the network input `p`, and the network object `net`, and returns the network outputs `a`.

### 4.6.1 Working Coding Of Simulation Data

```
load testData.mat
load myNN.mat
Y=sim(net, data10Mts);
Y=full(compet(Y))
[C, Rate]=confmat(Tts, Y)
```

## 4.7 Main Stuffing of System Process

1. `trainData.m` → create training data : `M T` → `trainData.mat`
2. `testData.m` → create testing data : `Mts Tts` → `testData.mat`
3. `valData.m` → create validation data : `Mval Tval` → `valData.mat`
4. `trainNN.m` → create network :  
    `load trainData.mat`  
    `load validationData.mat`  
    `net` → `myNN.mat`
5. `simData.m` → create simulation data :  
    `load testData.mat`  
    `load myNN.mat`  
    → `sim(net, dataMts)`

## 4.8 Result

Training Algorithms	Result
(Without validation) Trainlm { tansig,purelin}	1) Epochs: 1000 but stopped by the user at 125th epochs because of warning. Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 2.450757e-019.  In trainlm at 318 In network.train at 278 In trainNN at 20  <b>Percentage Of Accuracy:</b> Rate = 92 138  2) Epoch: 300, stopped at epoch 160 <sup>th</sup>  <b>Percentage Of Accuracy:</b> Rate = 100 150
Traingdm { logsig,logsig}	Epochs: 3000  <b>Percentage Of Accuracy:</b> Rate =6.6667 10.0000
Traingdm { tansig,purelin }	Epochs 1000  <b>Percentage Of Accuracy:</b> Rate = 0 0
Traingd { tansig,purelin }	Epochs: 1000, stopped at 6 <sup>th</sup> epoch  Neuron hidden layer: 10  <b>Percentage Of Accuracy:</b> Rate = 0 0
Traingdx { tansig,purelin}	Epochs: 1000,stopped at 19 <sup>th</sup>  <b>Percentage Of Accuracy:</b> Rate = 6.6667 10.0000
Trainrp { tansig,purelin}	[10 15]; netO.trainParam.epochs = 1000;  <b>Percentage Of Accuracy:</b> Rate =13.3333 20.0000

Training Algorithms	Result
Traincgf { tansig,purelin}	[10 15];netn.trainParam.epochs = 1000 <b>Percentage Of Accuracy:</b> Rate = 76.6667 115.0000
Traincgp { tansig,purelin}	[10 15]; netm.trainParam.epochs = 1000; <b>Percentage Of Accuracy:</b> Rate = 93.3333 140.0000
Trainlm { tansig,purelin}	[5 15] <b>Percentage Of Accuracy :</b> Rate = 72.6667 109.0000
Trainbfg { tansig,purelin}	[10 15]; epochs used: 1000 but epochs stopped at 18 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate = 13.3333 20.0000
Trainbfg { tansig,purelin}	[15 15]; epochs used: 500 but epochs stopped at 89 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate = 26.6667 40.0000
Trainbr { tansig,purelin}	[15 15]; epochs used: 1000 but epochs stopped at 37 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate =100 150
Traincgb { tansig,purelin}	[10 15]; epochs used: 1000 but epochs stopped at 864 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate = 99.3333 149.0000
Traingda { logsig,logsig }	[15 15]; Epochs: 1000, stopped 86 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate= 65.3333 98.0000
Trainoss { tansig,purelin}	[10 15]; Epochs: 1000, stopped 29 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate= 13.3333 20.0000
Trainoss { tansig,purelin}	[15 15]; Epochs: 1000, stopped 834 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate= 100 150
Trainscg { tansig,purelin}	[10 15]; Epochs: 1000, stopped 542 <sup>th</sup> <b>Percentage Of Accuracy :</b> Rate= 100 150

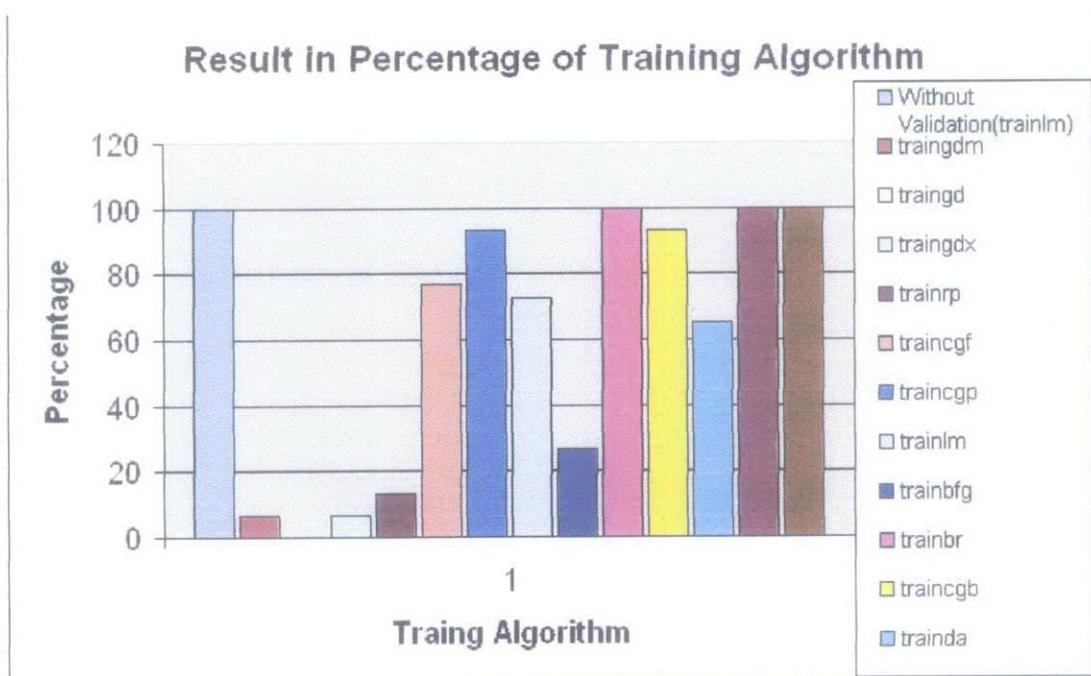


Figure 11: Graph of Training Algorithm Result

#### 4.9 Setting up of Graphical User Interface (GUI)

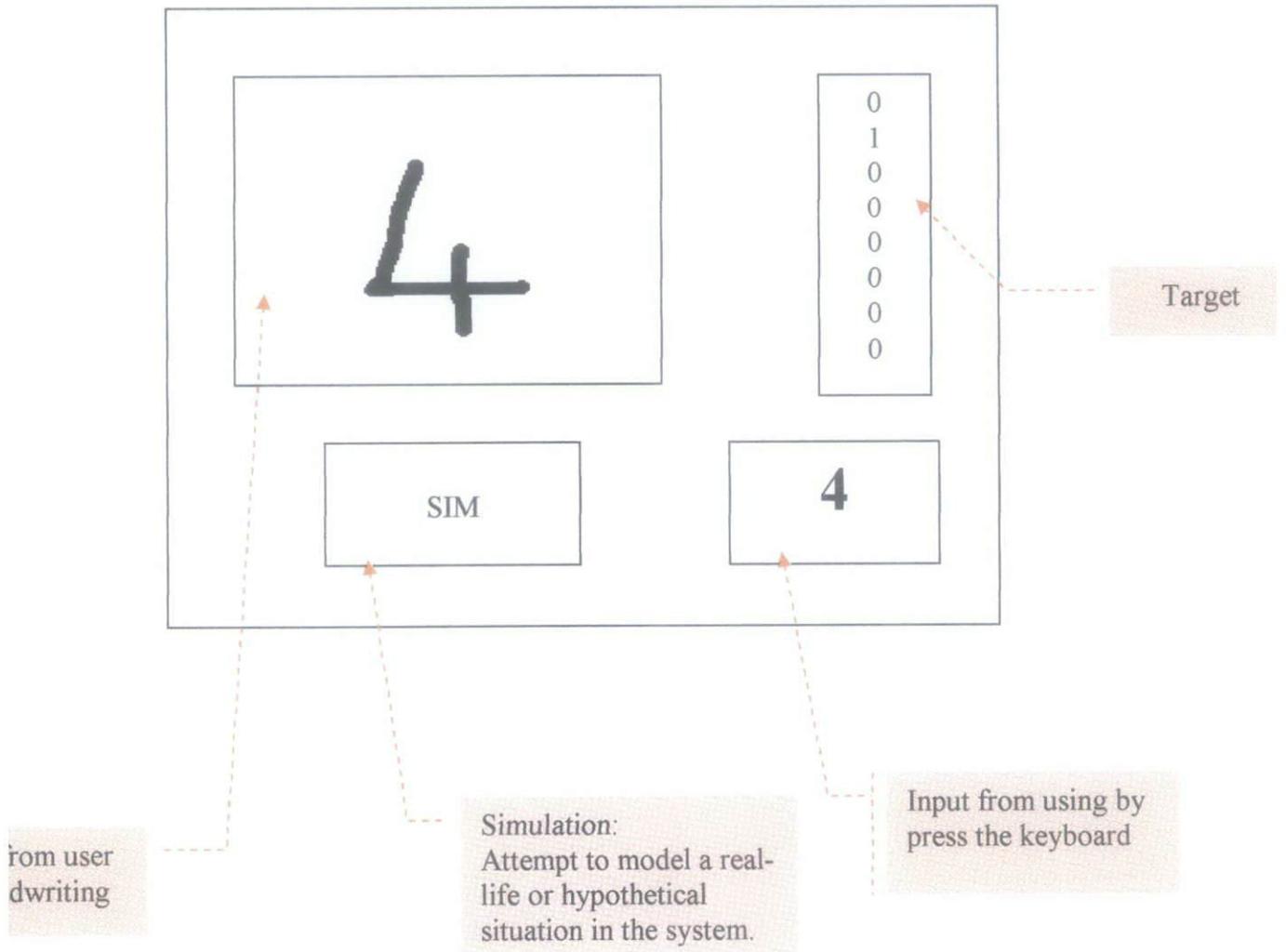


Figure 12: Graphical User Interface

#### 4.10 Idea of E-Handrawn's GUI Functionality

When inputs and targets are different sizes, system will convert the size of image number into standard size (100 x 100 pixel), if the size of image number too large (eg.1000x1000), system will shrink that number into standard size. But if that number too small, system will zoom the number to transform into standard size.

E-Handrawn's GUI can capture or recognize by each number until user write down the mathematical symbol such as plus sign, system proficient to recognize that number in a group. When user insert equal sign (=), system will stop to capture number and start doing the calculation function and give the particular answer.

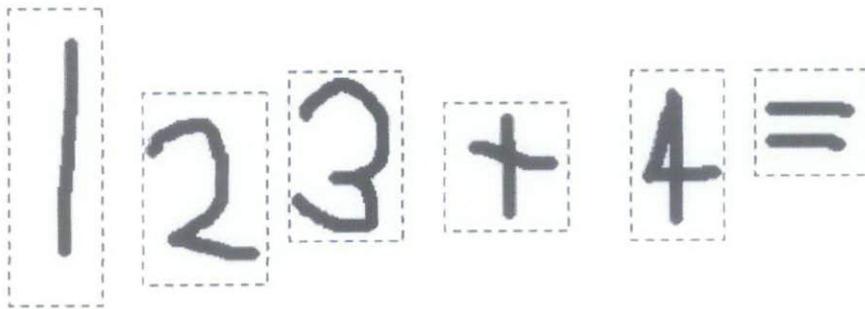


Figure 13: Picture or input before system resize

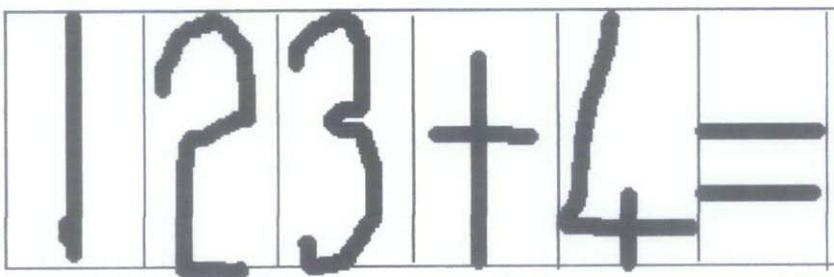


Figure 14: Picture or input after system resized

## CHAPTER 5

### CONCLUSION AND RECOMMENDATIONS

#### 5.1 Conclusion

To meet all the objectives of this system enhancement, a lot of effort must be put on. During this beginning phase, most of the activities are focus on understanding what neural network, backpropagation, pattern recognition and MATLAB are, how to training the system, testing the system, data validation, creating network object and data simulation. From my research; I achieved to find the best training algorithm, transfer function and network hidden layer; the results are below:

1. Highest performance of training algorithm:
  - trainbr 100% of accuracy
  - trainscg 100% of accuracy
2. The best combination of hidden layer is: [10 15] and [15 15]
3. The best combination of transfer function is: {tansig, purelin}

The purposes of this system are to develop a hand-writing calculator as alternative calculator in computer application and to develop a calculator that approach human natural way of calculation. The Final Year Project's concern is to develop a functioning system that can be use and further can be apply for livings. Literature reviews and theories have been refined in order to get more understanding by

reviewing books and internet pertaining on the Neural Network fundamental and MATLAB fundamental that related to the e-hand-drawn system.

## **5.2 Recommendations**

- Develop hand-drawing calculator as alternative calculator in computer application.
- To develop a calculator that more human natural way of calculation.
- Later on this project can be integrated in electronic paper (e-paper) or Digital Paper since nowadays its application is inadequate.

## CHAPTER 6

### REFERENCES

- i. [http://en.wikipedia.org/wiki/Where\\_Mathematics\\_Comes\\_From](http://en.wikipedia.org/wiki/Where_Mathematics_Comes_From)
- ii. [http://en.wikipedia.org/wiki/Neural\\_network](http://en.wikipedia.org/wiki/Neural_network)
- iii. <http://en.wikipedia.org/wiki/Perceptron>
- iv. <http://www.ccs.neu.edu/home/cash21/sigplan-calculator.pdf>
- v. Arbib, Michael A. (Ed.) (1995). *The Handbook of Brain Theory and Neural Networks*.
- vi. Mandic, D. & Chambers, J. (2001). *Recurrent Neural Networks for Prediction: Architectures, Learning algorithms and Stability*. Wiley.
- vii. Michael Negnevitsky (2002). *Artificial Intelligence : A Guide to Intelligent Systems*.
- viii. Widrow, B., Lehr, M.A., "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. IEEE*, vol 78, no 9, pp. 1415-1442, (1990).
- ix. <http://www.cs.bgu.ac.il/~omri/Perceptron/>
- x. *Neural Network Fundamentals & Implementation in MATLAB; Professional Training Course*, Activemedia Innovation SDN BHD.
- xi. <http://www.informingscience.org/proceedings/InSITE2005/I61f107Sala.pdf>
- xii. [http://en.wikiversity.org/wiki/Learning\\_and\\_Neural\\_Networks](http://en.wikiversity.org/wiki/Learning_and_Neural_Networks)

- xiii. <http://en.wikipedia.org/wiki/Simulation>
- xiv. [http://en.wikipedia.org/wiki/Digital\\_paper](http://en.wikipedia.org/wiki/Digital_paper)
- xv. [http://en.wikipedia.org/wiki/Electronic\\_paper](http://en.wikipedia.org/wiki/Electronic_paper)
- xvi. <http://docs.lib.purdue.edu/ecetr/275/>
- xvii. <http://www.freepatentsonline.com/5369622.html>
- xviii. <http://en.wikipedia.org/wiki/Preprocessing>
- xix. <http://en.wikipedia.org/wiki/MATLAB>
- xx. <http://en.wikipedia.org/wiki/Simulink>
- xxi. <http://www.mathworks.com/products/simpower/>
- xxii. <http://www.math.ufl.edu/help/matlab-tutorial/>

# CHAPTER 7

## APPENDIX

### 7.1 Appendix A

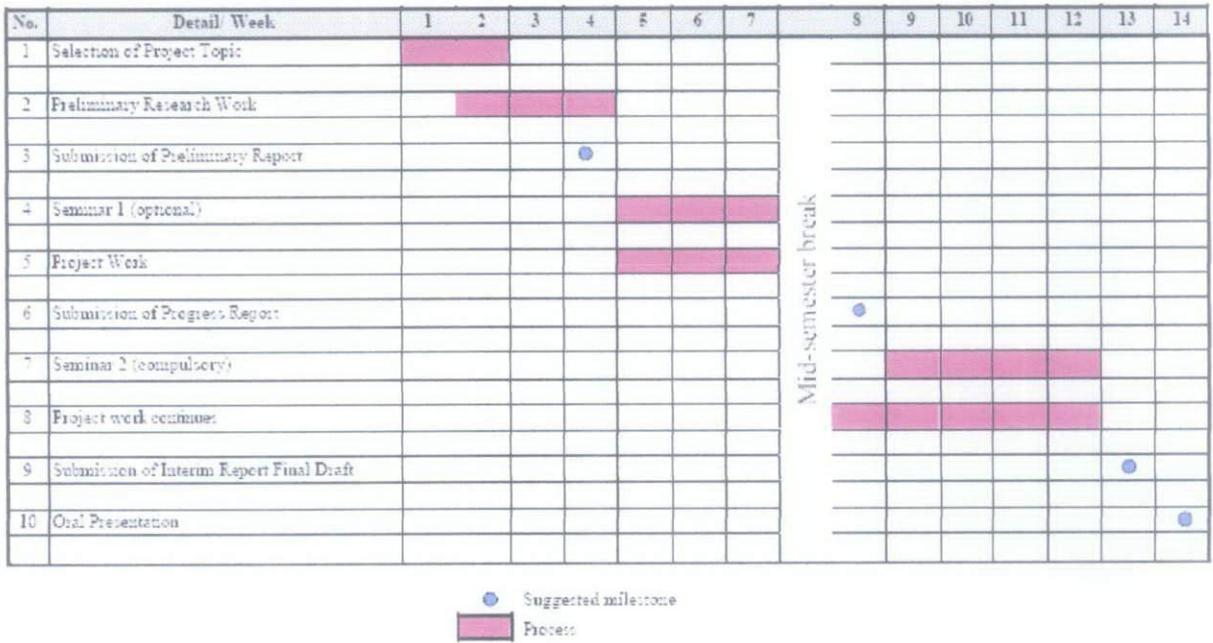
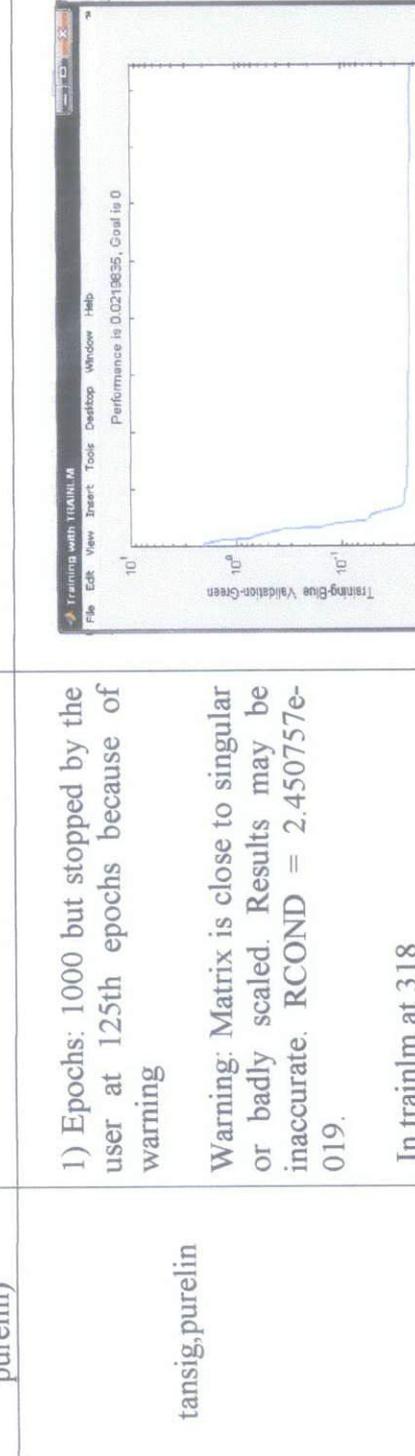


Figure 15: Milestone for the First Semester of 2 -Semester Final Year Project

Training Algorithms		Neuron Model (tansig, logsig, purelin)	Result
			Graph
Without validation trainlm	tansig, purelin	<p>1) Epochs: 1000 but stopped by the user at 125th epochs because of warning</p> <p>Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 2.450757e-019.</p> <p>In trainlm at 318 In network.train at 278 In trainNN at 20</p> <p>PERCENTAGE OF ACCURACY: Rate = 92 138</p> <p>2) Epoch: 300, stopped at epoch 160<sup>th</sup></p> <p>PERCENTAGE OF ACCURACY: Rate = 100 150</p>	 <p>The left screenshot shows a training graph for 'trainlm' with a y-axis from <math>10^{-2}</math> to <math>10^1</math> and an x-axis from 0 to 160 epochs. The performance curve shows a sharp increase starting around epoch 100, reaching a peak near <math>10^1</math> by epoch 160. The right screenshot shows a training graph for 'trainNN' with a y-axis from <math>10^0</math> to <math>10^1</math> and an x-axis from 0 to 3000 epochs. The performance curve shows a sharp increase starting around epoch 1500, reaching a peak near <math>10^1</math> by epoch 3000.</p>

		Result	
		Remarks	Graph
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Epochs 3000 Rate = 6.6667 10.0000	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
traingd	tansig, purelin	<p>Epochs: 1000, stopped at 6<sup>th</sup> epoch</p> <p>Neuron hidden layer: 10</p> <p>Percentage:</p> <p>Rate = 0 0</p>	<p>Performance is 2.81427, Goal is 0</p>

		Result	
		Remarks	Graph
Training Algorithms	Neuron Model (tansig, logsig, purelin)		
Traingdx	tansig,purelin	<p>Epochs: 1000, stopped at 19<sup>th</sup></p> <p>Rate = 6.6667 10.0000</p>	

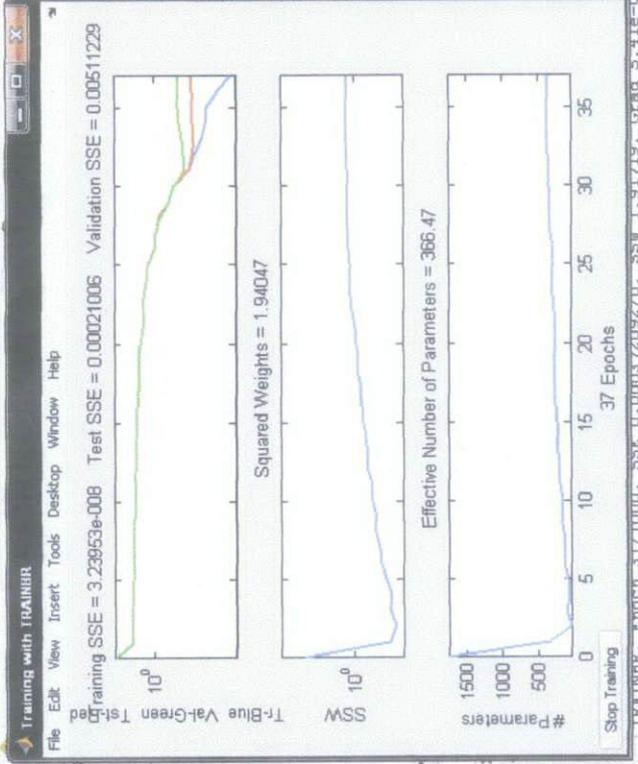
		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
trainrp	tansig,purelin	<pre>[10 15], {'tansig','purelin'}, 'trainrp') netO.trainParam.epochs = 1000; Rate = 13.3333 20.0000</pre>	

		Result	
		Remarks	Graph
Training Algorithms	Neuron Model (tansig, logsig, purelin)		
traincgf	tansig, purelin	<pre>[10 15] {'tansig','purelin'} netn.trainParam.epochs = 1000; Rate = 76.6667 115.0000</pre>	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
traincgp	tansig, purelin	<pre>[10 15]{'tansig','purelin'} netm.trainParam.epochs = 1000; Rate = 93.3333 140.0000</pre>	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
trainlm	tansig,purelin	[5 15] Rate = 72.6667 109.0000	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
Trainbfg	tansig, purelin	<pre>[15 15] {'tansig', 'purelin'}, 'trainbfg') netj.trainParam.show = 1; netj.trainParam.epochs = 500; Rate = 26.6667 40.0000</pre>	

Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Result
trainbr	Tansig,purelin	<pre>netg.trainParam.show = 1; netg.trainParam.epochs = 1000; Rate =     100    150</pre>	<p style="text-align: center;">Graph</p>  <p>The screenshot shows the 'Training with TRAINER' window with the following data:</p> <ul style="list-style-type: none"> <li>Training SSE = 3.23853e-008</li> <li>Test SSE = 0.00021006</li> <li>Validation SSE = 0.00511229</li> <li>Squared Weights = 1.94047</li> <li>Effective Number of Parameters = 366.47</li> <li>Epochs: 37</li> </ul>

		Result	
		Remarks	Graph
Training Algorithms	Neuron Model (tansig, logsig, purelin)		
traincgb	Tansig, purelin	<pre>[10 15] {'tansig','purelin'}, 'traincgb') net1.trainParam.show = 1; net1.trainParam.epochs = 1000; Rate = 99.3333 149.0000</pre>	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
traingda	logsig,logsig	<p>Epochs: 1000, stopped 86<sup>th</sup> [15 15] {'logsig','logsig'},'traingda') Rate = 65.3333 98.0000</p>	

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
trainoss	'tansig', 'purelin'	<pre>[15 15] {'tansig','purelin'}, 'trainoss') neth.trainParam.show = 1; neth.trainParam.epochs = 1000; Rate = 100 150</pre>	

		Result	
		Remarks	Graph
Training Algorithms	trainscg	<pre>[10 15] {'tansig','purelin'},'trainscg') netk.trainParam.show = 1; netk.trainParam.epochs = 1000; Rate = 100 150</pre>	<p>Performance is 0.0252804, Goal is 0</p> <p>542 Epochs</p> <p>TRAINSCG, Epoch 542/1000, MSE 0.0252797, Gradient 0.1430697E+000</p>

		Result	
Training Algorithms	Neuron Model (tansig, logsig, purelin)	Remarks	Graph
trainbr	'tansig','purelin'	<pre>[15 15] {'tansig','purelin'}, 'trainbr'</pre> <p>Epochs: 1000, stopped 38<sup>th</sup></p> <p>Rate = 100 150</p>	<p>The graph displays the training process for a neural network using the 'trainbr' algorithm. It consists of three subplots sharing a common x-axis representing epochs from 0 to 38.</p> <ul style="list-style-type: none"> <li><b>Top Graph (Training SSE):</b> Shows the training Sum of Squared Errors (SSE) on a logarithmic scale (10<sup>0</sup>). The training SSE (green line) decreases from approximately 10<sup>0</sup> to about 10<sup>-1</sup> by epoch 38. The test SSE (red line) and validation SSE (blue line) also decrease, with the validation SSE reaching approximately 0.00343527.</li> <li><b>Middle Graph (Squared Weights):</b> Shows the Squared Weights (SSW) on a logarithmic scale (10<sup>0</sup>). The SSW (blue line) remains relatively constant around 10<sup>0</sup> throughout the training process.</li> <li><b>Bottom Graph (Effective Number of Parameters):</b> Shows the Effective Number of Parameters (blue line) on a linear scale from 0 to 1500. The number of parameters starts at approximately 1500 and decreases to about 405.916 by epoch 38.</li> </ul> <p>Summary statistics from the graphs:      Training SSE = 1.97505e-007    Test SSE = 2.04436e-005    Validation SSE = 0.00343527      Squared Weights = 1.63517      Effective Number of Parameters = 405.916</p>