

# **HANDWRITING RECOGNITION USING WEBCAM FOR DATA ENTRY**

By

**WONG YOONG XIANG**

14312

Final Report submitted in partial fulfilment of  
the requirements for the  
Degree Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

FYP II September 2014

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

**MR PATRICK SEBASTIAN**

# **CERTIFICATION OF APPROVAL**

## **HANDWRITING RECOGNITION USING WEBCAM FOR DATA ENTRY**

by

Wong Yoong Xiang

A project dissertation submitted to the  
Electrical & Electronics Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfilment of the requirement for the  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Approved:

---

Mr. Patrick Sebastian  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK

September 2014

## **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.

---

Wong Yoong Xiang

## ABSTRACT

The Handwriting Recognition using Webcam for Data Entry project has its primary purpose to develop a system or algorithm that is robust enough to recognize numerical handwritings. A web camera is to be utilized to capture images of handwritten scores and question numbers on the examination score sheet in real time. It is then preprocessed and all the features are being fed into a neural network that is already been trained by various test samples. The outcome of the project should be able to obtain a system that is able to recognize handwritten numerical data with the lowest overshoot and errors. Several distinctive feature from each character is extracted using a few feature extraction methods, in which a comparison between three types of feature extraction modules were used. The first test was done with a neural network trained with only the Character Vector Module as its feature extraction method. A result that is far below the set point of the recognition accuracy was achieved, a mere average of 64.67% accuracy. However, the testing were later enhanced with another feature extraction module, which consists of the combination of Character Vector Module, Kirsch Edge Detection Module, Alphabet Profile Feature Extraction Module, Modified Character Module and Image Compression Module. The modules have its distinct characteristics which is trained using the Back-Propagation algorithm to cluster the pattern recognition capabilities among different samples of handwriting. Several untrained samples of numerical handwritten data were obtained at random from various people to be tested with the program. The second tests shows far greater results compared to the first test, have yielded an average of 84.52% accuracy. As the recognition results have not reached the target of 90%, further feature extraction modules are being recommended and an additional feature extraction module was added for the third test, which successfully yields 90.67%. With the time-frame target achieved, a robust data entry system was developed using web camera together with a user-friendly GUI (Graphical User Interface).

## ACKNOWLEDGEMENTS

With immense pleasure, it is my greatest gratitude to be blessed with the opportunity to undertake Final Year Project in Universiti Teknologi PETRONAS (UTP). This Final Year Project report, entitled ‘Handwriting Recognition using Webcam for Data Entry’, was the outcome of vast hard work, effort placed and is completed in a span of 2 active semesters. Reminiscing the process of project completion, the author is appreciative towards numerous individuals whom are accountable for their assistance towards achieving the project’s objective.

Primarily, the author would like to place on record, a sincere gratitude to Mr. Patrick Sebastian, the project supervisor, whom have trusted the author on his capabilities and competencies on handling the project towards completion. Without Mr. Patrick Sebastian’s selfless knowledge, support, advice and guidance, the project would not be completed on time.

Additionally, this Final Year Project has successfully gained recognition and reputation in the 34<sup>th</sup> Science and Engineering Design Exhibition (SEDEX34) held in Universiti Teknologi PETRONAS (UTP) on the 8<sup>th</sup> and 9<sup>th</sup> of December 2014, whereby it won the Gold Award for Electrical and Electronics Engineering Final Year Project Category and Idea Generation Funds for Education worth RM10,000.00 (Appendix J).

Subsequently, I would also like to express special and sincere thanks to all colleagues, classmates and seniors for all the responsive help, encouragements and guidance towards the accomplishment of the project. Last but not least, I also placed on record, his sense of gratitude towards his parents for their unceasing encouragement and support. Superior thanks also to one and all who, directly or indirectly, have lent their helping hand in contributing towards the success of this project.

## TABLE OF CONTENTS

ABSTRACT .....	iv
ACKNOWLEDGEMENTS .....	v
TABLE OF CONTENTS .....	vi
LIST OF FIGURES .....	ix
LIST OF TABLES .....	xi
LIST OF ABBREVIATIONS .....	xii
CHAPTER 1: INTRODUCTION .....	1
1.1 BACKGROUND .....	1
1.2 PROBLEM STATEMENT .....	2
1.3 OBJECTIVES .....	2
1.4 SCOPE OF STUDY .....	3
CHAPTER 2: LITERATURE REVIEW AND THEORY .....	4
2.1 BACK PROPAGATION NEURAL NETWORK .....	4
2.2 IMAGE PREPROCESSING & FEATURE EXTRACTION .....	6
2.2.1 MODIFIED ALPHABETS ENCODER MODULE .....	7
2.2.2 EDGE DETECTION METHOD .....	8
2.2.3 KIRSCH EDGE DETECTION MODULE .....	8
2.2.4 IMAGE COMPRESSION MODULE .....	10
2.2.5 CHARACTER VECTOR MODULE .....	10
2.2.6 CURVATURE VECTOR MODULE .....	10
2.3 CHARACTER RECOGNITION .....	11
CHAPTER 3: METHODOLOGY .....	12
3.1 METHODOLOGY .....	12
3.2 SOFTWARE & HARDWARE TOOLS .....	13
3.3 IMAGE PRE-PROCESSING .....	13

3.3.1	IMAGE NOISE REDUCTION.....	15
3.3.2	SCRATCHES REMOVAL.....	16
3.3.3	SLANT CORRECTION .....	17
3.4	FEATURE EXTRACTION MODULES .....	18
3.4.1	CHARACTER VECTOR MODULE .....	20
3.4.2	KIRSCH EDGE DETECTION MODULE.....	20
3.4.3	ALPHABET PROFILE EXTRACTION MODULE .....	21
3.4.4	MODIFIED CHARACTER MODULE.....	22
3.4.5	IMAGE COMPRESSION MODULE .....	23
3.4.6	CURVATURE VECTOR MODULE .....	24
3.5	NEURAL NETWORK.....	26
3.6	IMAGE ACQUISITION MODULE.....	28
3.6.1	PROTOTYPE.....	29
3.6.2	ACQUIRING THE REGION OF INTEREST .....	31
3.7	GRAPHICAL USER INTERFACE MODULE.....	32
3.7.1	GRAPHICAL USER INTERFACE COMMAND STEPS.....	33
3.8	GANTT CHART & KEY MILESTONES .....	39
CHAPTER 4: RESULTS AND DISCUSSION .....		41
4.1	HANDWRITING RECOGNITION TESTS.....	41
4.1.1	TEST 1 .....	41
4.1.2	TEST 2 .....	43
4.1.3	TEST 3 .....	47
4.1.4	COMPARISON AMONG ALL THREE TESTS .....	50
4.1.5	COMPARISON AMONGST FEATURE EXTRACTION MODULES ...	51
CHAPTER 5: CONCLUSION AND RECOMMENDATIONS .....		52

REFERENCES.....53

APPENDICES .....56

    APPENDIX A: TEST 1 MATLAB CODES .....56

    APPENDIX B: TEST 2 MATLAB CODES.....60

    APPENDIX C: TEST 3 MATLAB CODES.....70

    APPENDIX D: TEST 1 TRAINING SET .....73

    APPENDIX E: TEST 2 TRAINING SET .....74

    APPENDIX F: TEST 3 TRAINING SET.....75

    APPENDIX G: TEST SET SAMPLE FOR TEST 1 .....76

    APPENDIX H: TEST SET SAMPLE FOR TEST 2 .....78

    APPENDIX I: TEST SET SAMPLE FOR TEST 3.....80

    APPENDIX J: SEDEX AWARDS .....84



## LIST OF FIGURES

Figure 1: Modified Alphabet Encoder Classified Regions .....	7
Figure 2: Edge Detection Method Classified Regions .....	8
Figure 3: Predefined masks of Kirsch Edge Detection Module.....	9
Figure 4: Example of Kirsch Edge Detection Module output.....	9
Figure 5: Character Vector Module Elements .....	10
Figure 6: Process flow identification and implementations for the handwritten recognition project.....	12
Figure 7: Process flow of pre-processing raw image using MATLAB's Image Processing Toolbox .....	14
Figure 8: Image after object plotting which detects the unwanted noise .....	15
Figure 9: Process Flow Image Noise Reduction .....	15
Figure 10: Scratches removal.....	16
Figure 11: Numbers before and after slant correction .....	17
Figure 12: Process Flow of Character Vector Module Algorithm.....	20
Figure 13: Process Flow of Kirsch Edge Detection Module Algorithm.....	21
Figure 14: Process Flow of Alphabet Profile Feature Extraction Module Algorithm .....	22
Figure 15: Process Flow of Modified Character Module Algorithm.....	23
Figure 16: Process Flow of Image Compression Module Algorithm .....	24
Figure 17: Process Flow of Curvature Vector Module Algorithm .....	25
Figure 18: Output from first camera .....	28
Figure 19: Output from second camera.....	29
Figure 20: Location of the camera and the prototype .....	30
Figure 21: Defining the region of interest.....	31
Figure 22: Graphical User Interface Layout for Handwriting Recognition System using Webcam for Data Entry.....	32
Figure 23: GUI Command Steps.....	33
Figure 24: Extracting data from Webcam 1 onto the GUI.....	34
Figure 25: Extracting data from Webcam 2 onto the GUI.....	35
Figure 26: Comparing the total marks with sub marks on GUI.....	35

Figure 27: Verifying data entry on GUI.....36

Figure 28: Completed GUI .....37

Figure 29: Exported data to the Excel Database .....38

Figure 30: Exported data to the Excel Database .....38

Figure 31: Mistakes and occurrences of each number in Test 1 .....42

Figure 32: Percentage of accuracy of each number in Test 1 .....42

Figure 33: Mistakes and occurrences of each number in Sample 3 of Test 2.....44

Figure 34: Percentage of accuracy of each number in Sample 3 of Test 2.....45

Figure 35: Mistakes and occurrences of each number in Sample 4 of Test 2.....45

Figure 36: Percentage of accuracy of each number in Sample 4 of Test 2.....46

Figure 37: Graph of Number of Training Sets vs Avg. Accuracy .....48

Figure 38: Comparison between the percentages of accuracy between 3 tests.....50

## LIST OF TABLES

Table 1: Comparison between matrix sizes of two tests .....	19
Table 2: Neural Network Training Parameters .....	27
Table 3: Gantt Chart and Key Milestones for FYP I .....	39
Table 4: Gantt Chart and Key Milestones for FYP II .....	40
Table 5: Percentage of accuracy of Test 1 .....	41
Table 6: Percentage of accuracy of Test 2 .....	44
Table 7: Neural Network Training Accuracy.....	47
Table 8: Percentage of accuracy of Test 3 .....	49
Table 9: Comparison test with each feature extraction module .....	51

## **LIST OF ABBREVIATIONS**

A.I.: Artificial Intelligence

UTP: Universiti Teknologi PETRONAS

OCR: Optical Character Recognition

ANN: Artificial Neural Network.

MLP: Multilayer Perceptron

ROI: Region of Interest

GUI: Graphical User Interface

# CHAPTER 1: INTRODUCTION

## 1.1 BACKGROUND

Teachers and lecturers of various academia fields have been given the authority for many years to carry out data entry of student's academic results into computers. There is a need to reduce the burden of these educators with the enhancement of science and technology, which sees the advancements and implementations of artificial intelligence (A.I.) in various industries. Handwriting recognition systems have worked its way into the scientific era of modern technology but has yet to achieve a prime solution for various handwriting trends. Applications such as passport number recognition, signature identification and plate number scanning are the examples of optical character recognitions (OCR).

In Malaysia, general or public examinations are mandatory to be taken by all students, with educators responsible in conveying information such as marks and total scores from the examination score sheet to be computerized. Human errors are prone to arise, with data often entered incorrectly causing the student victim to be given undeserved grades for the particular examination. Thus, it is oblique to have a handwriting recognition system for current or future data entry into online systems, which will also facilitates the transfer of data across web link systems in the education industry.

Throughout the years, various efficient techniques have been deployed by researchers to recognize various numeric handwritten characters, but still remains a sturdy hurdle with thousands of different shaped handwriting trends. This project has high similarities from a previous thesis by Poo entitled, "Handwriting Recognition for Data Entry (HandRec)" [1], which in this project, the goal is to achieve a more robust output.

## **1.2 PROBLEM STATEMENT**

Humans have unique handwriting styles which proves to be an obstacle for handwriting recognition algorithms. To date, multiple researches have been done to recognize these different handwriting styles, most notable using the artificial neural network (ANN) with back propagation algorithms [2], which has also been proven to give adequately high accuracies. By using real time process image capturing, this system and algorithm can be implemented to apply multiple handwritten entry data for schools and universities, where the handwritten data of a standard score sheet from different individuals can be transferred to a spreadsheet.

## **1.3 OBJECTIVES**

The prime purpose of the project is to utilize a web camera to capture images of handwritten scores and question numbers on the examination score sheet, with an algorithm that has the capability to recognize handwritings and computerized numerals in real time. In short, an intelligent neural network has to be developed for robust handwriting recognition of the numbers from the webcam to be able to be input into a database. The neural network method will be deeply explored to obtain the optimal solution with the lowest overshoot and errors. Other methods of handwriting recognition includes Fuzzy Logic [3], Hidden Markov Model [4], Principal Component Analysis (PCA) [5], Local Affine Transformation (LAT) [6], Wavelet Transform [7] and Curvature Coefficient Method [8].

## **1.4 SCOPE OF STUDY**

The current range of study of the project is limited to only Universiti Teknologi PETRONAS (UTP) score sheet handwriting recognitions. Handwriting trend subjects will only be concentrated on various lecturers and students within UTP, in which a flexible image will be captured by a webcam and a robust neural network will be trained based on the samples obtained. The handwritten data of question number written by students and examination scores written by lecturers will only be taken into consideration for data entry into an online database system. Further studies and future researches would include handwritings by various people of various professions on different applications.

## CHAPTER 2: LITERATURE REVIEW AND THEORY

### 2.1 BACK PROPAGATION NEURAL NETWORK

Researchers have used different approaches towards solving the handwriting recognition problems, with some trying to fuse multiple recognition algorithms to overcome the weak points of another algorithm. Nonetheless, neural network as a single algorithm is robust enough to recognize handwriting samples and produce high recognition rate. Neural network has self-learning, self-adapt and self-process capabilities which make it robust for handwritten recognition systems [9]. There are three layers in the typical neural network, which are the input layer, output layer and the hidden layer. These layers correspond with each other to train training sets for recognition algorithms.

During handwritten recognition process, the pre-processed image is interpreted in the form of an input signal which is then propagated through the network of the three layers in forward direction. The neural network is sometimes referred as multilayer perceptron (MLP). One advantage of using neural network is that it can be trained to perform the error-correction learning rule [10].

Most of the research that is based on neural network have achieved ultimately high recognition accuracy of more than 95%. Even with this high accuracy, the correct learning rate ( $\mu$ ) has to be chosen to ensure the recognition results obtained in optimal. The learning rate is the rate of which the number of training sample sets used to train the neural network. If a low value of  $\mu$  is chosen, the result obtained will be slow and inaccurate. However, if a high value of  $\mu$  is chosen, the algorithm will memorize the training sets and exceed the threshold consistency of recognition [11].



Throughout the years, neural network has been recognized as the key for future computer processing improvisations, as it has proven to be very successful solving numerous problem domains, particularly in diverse areas of science and technology [1]. Neural network has been able to counter problems which requires the computer processor to predict and classify a control problem, instead of just following programming algorithms [1]. Some of the key factors which has contributed to the sweeping success of this systems are:

- **Computing capabilities:** Neural networks consists of extremely complex functions and various sophisticated modeling techniques. As its modeling is non-linear, as compared to various linear modeling techniques, it has the advantage of simulating functions to predict an outcome given an input that has already been trained throughout its neurons. One advantage of using neural network is that it could handle large amount of computing command and still perform optimally.
- **Ease of use:** Neural network is an intelligent system whereby it learns by training samples. It has the ability to gather input data and produce complex functions to automatically learn the set of data, as well as interpreting the outcomes. Neural network learn by various examples and does not require the user to select distinctive features of the training data set, as the system has the ability to produce analysis itself. Hence, it is highly regarded for this reason that the neural network is used to encounter the problem in this handwritten recognition project.
- **Convenience:** The neural network system also has now been widely used on different platforms. One such common platform is MATLAB, which already has a pre-installed Neural Network Toolbox for the convenience of the user. The neural network need not to be implemented using long and hefty coding, but only requires the user to understand the amount of training iterations and parameters to be used to train the given input data.

## 2.2 IMAGE PREPROCESSING & FEATURE EXTRACTION

Before the numeric character is being trained in the neural network, the sample of data must first be pre-processed to extract the relevant data and information to be trained. Among the common pre-processing and feature extraction used in general, is by binarization of a digitized image of handwritten samples and forming an array of data pixels [12].

There are several features of the handwritten in which must be considered to be trained and in this paper [13], it focused on a few features namely the Gradient Based Wavelet Features, MAT based Directional Features, Complex Wavelet Features, Binary Gradient Directional Features, Median Filter Gradient Features, Image Thinning Distance Feature and Geometrical Features. It is in contrast with the feature extraction module with [1], who uses the Horizontal, Vertical and Diagonal Alphabet Regions Encoder Features, Kirsch Edge Detector Feature, Image Compression Features and Profiling Alphabet Smoothness, Width and Height Features. Both these feature extraction methods can be compared based on the recognition accuracy. In fact, there are many other extraction feature models that are used by various researchers.

Through feature extraction modules, the important elements of the handwritten data can be trained to achieve the highest recognition rate. If some of the character written to be recognized has missing features, the neural network has the ability to respond to the error and discriminate patterns [14]. Researchers have also tried using multilayer feed forward neural network (MLFFNN) [15] and auto associative neural network [16], both which are upgraded versions of back propagation neural network and achieved a slightly higher percentage of accuracy. These methods differ by the number of input layers, hidden layers and output layers used. Despite insignificant differences shown, the accuracy of recognition increased slightly.

## 2.2.1 MODIFIED ALPHABETS ENCODER MODULE

This feature extraction module uses 15 different regions to map out the number of bits of the particular handwritten number. In order to achieve equal and identical marked bit regions, the image is rescaled first onto 32 x 32 bitmap, which is then divided into 15 different regions, consisting 8 vertical regions (V1, V2, V3, V4, V5, V6, V7 and V8), 4 diagonal regions (D1, D2, D3 and D4) and 3 horizontal regions (H1, H2 and H3) as shown in Figure 1.

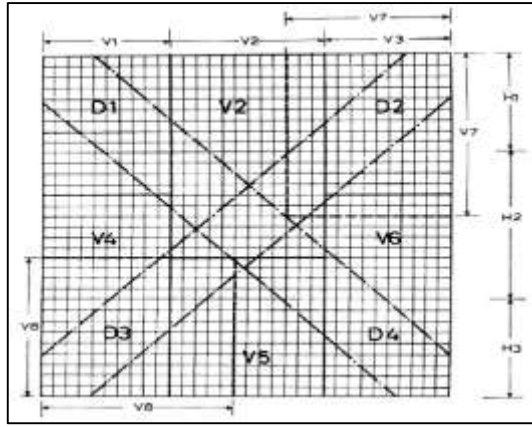


Figure 1: Modified Alphabet Encoder Classified Regions

Once the featured region has been divided, the number of marked bits is taken into account and computed using Equation 1, where the region feature is the sum of marked bits divided by the region size.

$$\text{region feature} = \frac{\text{sum of marked bits}}{\text{region size}} \quad (\text{Equation 1})$$

### 2.2.2 EDGE DETECTION METHOD

This feature extraction module involved the detection of the other edge of each number character from four different corner views, namely from the top, right, bottom and left direction. As depicted in Figure 2, the number of bits from each direction is taken into consideration with the program running through from each specific direction. Once a bit change is detected, the edge of the number character from each direction is computed and the extraction of features from the number is completed.

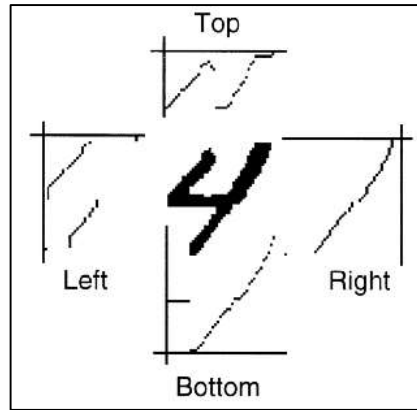


Figure 2: Edge Detection Method Classified Regions

### 2.2.3 KIRSCH EDGE DETECTION MODULE

This feature extraction module is able to distinguish the unique features of each of the 10 numerical digits. 8 predefined masks or filters were first determined as shown in Figure 3, which is then used to detect the horizontal, vertical, right – diagonal and left – diagonal edge of the number. Next, convolution between the 8 defined masks and the binary alphabet image takes place to produce 8 Kirsch Edge Detection Module images, 2 of each direction each.

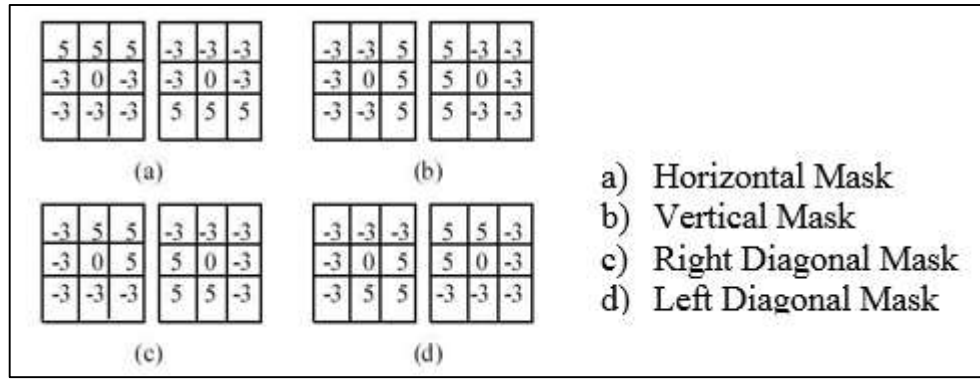


Figure 3: Predefined masks of Kirsch Edge Detection Module

All the 8 Kirsch Edge Detection Module images will be divided into pairs and will undertake a process of determining the highest number of bit values from each of the pair images. The output of the module as shown as an example in Figure 4 is that there will be one image from each defined direction, namely horizontal, vertical, right – diagonal and left – diagonal. The images will be compressed before being future processed.

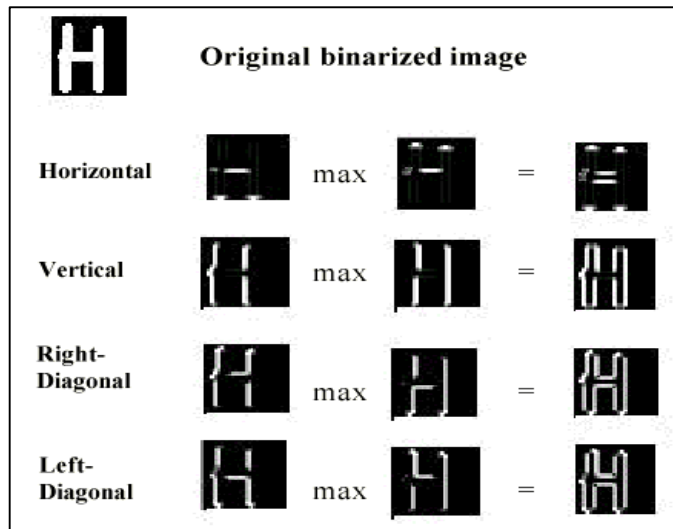


Figure 4: Example of Kirsch Edge Detection Module output

## 2.2.4 IMAGE COMPRESSION MODULE

The purpose of this feature extraction module is to reduce the computing power needed for image processing, as well as obtaining the features from a compressed image. The size of the image cropped from pre-processing of the raw image is reduced by 16 times using Equation 2 which will give a different result of feature extraction module.

$$T(i, j) = \frac{\sum_{x=2i-1}^{2i} \sum_{y=2j-1}^{2j} S(x, y)}{4} \quad (\text{Equation 2})$$

## 2.2.5 CHARACTER VECTOR MODULE

This feature extraction module is the most simple extraction module of image processing of characters. The single character is simply segmented into 35 equal areas of 5 x 7. The matrix of each element will represent the ratio of marked bits area over the ration of unmarked bits area in these 35 areas, as shown in Figure 5.

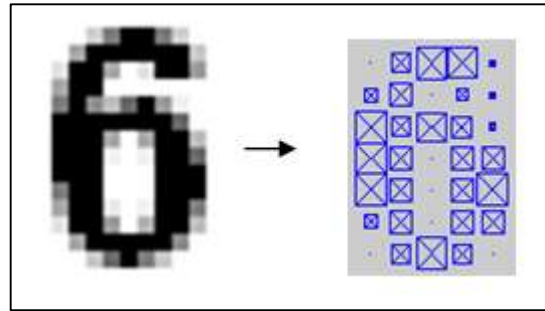


Figure 5: Character Vector Module Elements

## 2.2.6 CURVATURE VECTOR MODULE

This feature extraction module is another method to extract important data out from numerical handwritten data, in which the numbers can be differentiated through curves or circles. The module calculates the number of bits which forms a curve or circle at two different regions, top half and bottom half of the number.

## 2.3 CHARACTER RECOGNITION

Character recognition has not much difference compared to word recognition, but easier to be classified and need not to be segmented into individual characters. In this paper, character recognition is only focused towards producing a robust individual numerical handwritten numbers recognition system instead of grouped numbers, due to its complexity and time constraint.

Various researches have been carried out to distinguish handwritten data into digital form. One such method is through determining the distinctiveness and similarities which are present in the handwritten numerals as being carried out by [17]. This paper highly emphasized on the consideration of placing crucial parts into crucial combinations of numeral handwritten data features to exploit better recognition rate. This method was highly regarded as success, as various handwritten data patterns are being considered. The paper considered 29 patterns of 10 numerals with each giving a different but above 89% of mean percentage accuracy.

As compared to [18], the paper has carried out several experiments and proposed a few different methods to recognize unconstrained handwritten numerals. It is mentioned that a good feature extraction method should represent the numbers 0 to 9, and are able to distinguish the unique feature in each of them. Kirsch masks were proposed as being used in [1] recognition system as well, and are able to obtain an average of 97% accuracy based on the training set used.

From the above research papers, it is found that by occupying and combining a few methods to extract information from each handwritten numerals could greatly improve the handwritten recognition accuracy rate. Another method that could be considered in improving handwritten recognition accuracy is through statistical method which involves knowing the distinct features of the numbers such as circles, nodes and intersections in between the handwritten numbers.

## CHAPTER 3: METHODOLOGY

### 3.1 METHODOLOGY

Based on the reviews of various handwriting recognition techniques from literature reviews, artificial neural network is the choice of implementation for this project, in which MATLAB software will be used. Neural network is seen to be the best and wisest choice, as it has been proven to give the best results and accuracy.

Figure 6 shows the process flow identification and implementations for the handwritten recognition project using neural network.

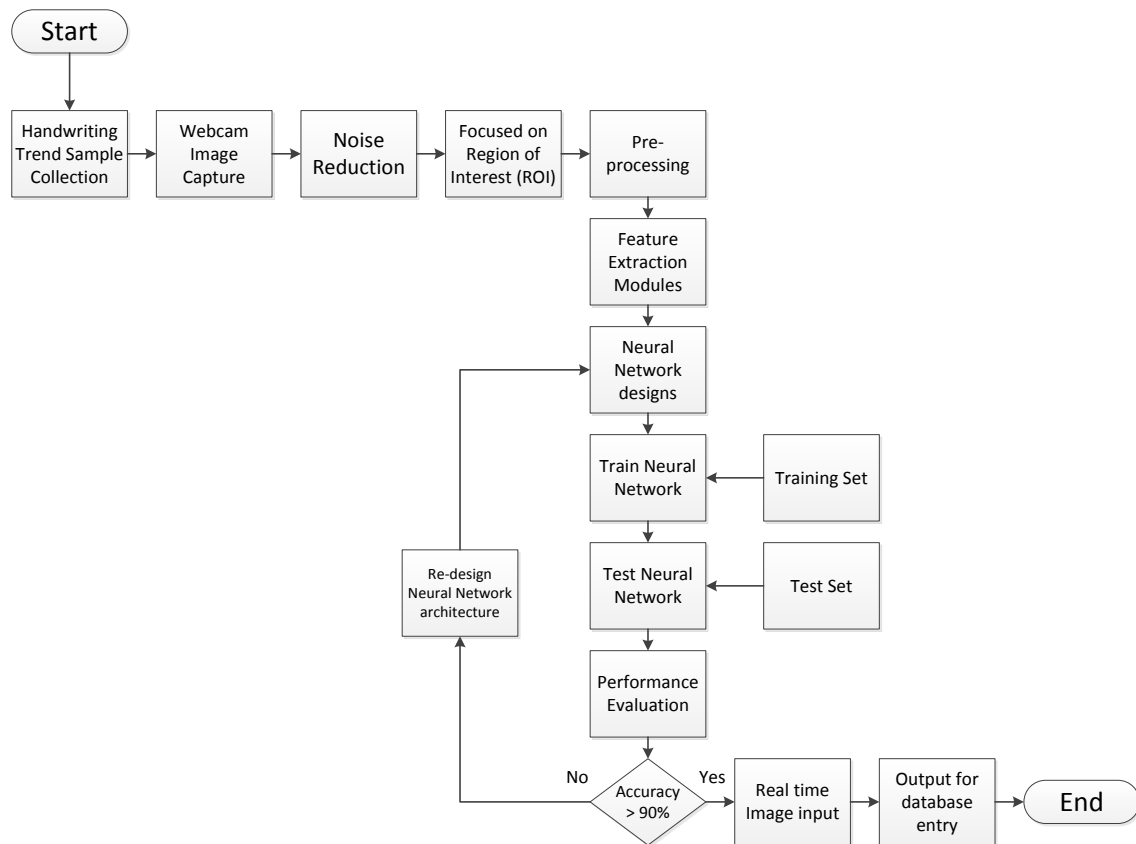


Figure 6: Process flow identification and implementations for the handwritten recognition project



## **3.2 SOFTWARE & HARDWARE TOOLS**

In this early stages of implementation, a scanner will be used to entry handwritten data onto MATLAB to be trained and tested for its accuracy. Once the neural network handwritten recognition system has been developed maturely, a web camera will be used for real time image input and algorithms have to be implemented to capture the data in the region of interest (ROI). The information captured, as well as the recognized handwritten data will be stored in a database which can easily be accessed through a Graphical User Interface (GUI).

## **3.3 IMAGE PRE-PROCESSING**

When an image is fed into the MATLAB handwriting recognition system, either from a scanner or a web camera, it is vital to process the image using standard signal-processing techniques for easy and appropriate data acquisitions. Noises are the most common portion of the image that have to be discriminated and removed. There are two types of noises defined in this project. One of it is insignificant, and could disorientate the recognition accuracy, usually small dots, ticks or particles which does need to be recognized as handwritten data. The other is scratches of handwritten data which should be discounted, which usually forms a larger number of bits compared to normal handwritten data. The following pre-processing techniques are used to remove noise and extract individual numerical handwritten data in MATLAB (Figure 7).

The MATLAB software has already the Image Processing Toolbox which has the capabilities to perform ideal image processing using the matrix data structure [19]. Various functions within the toolbox has been explored by researchers such as [11], together with MATLAB's Neural Network Toolbox on the basis of training data for handwriting recognition programs. Yet, the methods used varies with different programmers.

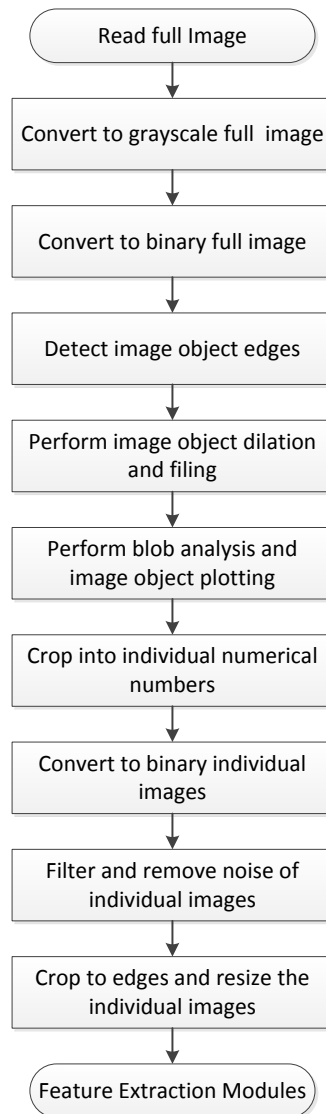


Figure 7: Process flow of pre-processing raw image using MATLAB's Image Processing Toolbox

### 3.3.1 IMAGE NOISE REDUCTION

In this first phase of a robust handwriting algorithm implementation, scanned images were used, which would later be turned into binary format images. It is known that little distortions such as unwanted markings or paper cripple can cause noise to the image processing module to detect the noise as one of its individual object (Figure 8). These noises have to be removed from the system before being transferred into the Feature Extraction Modules for further processing.

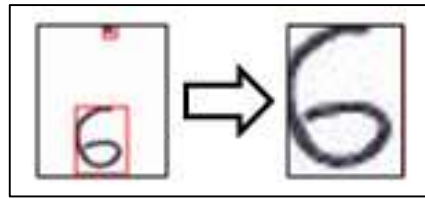


Figure 8: Image after object plotting which detects the unwanted noise

A simple method was implemented by summing all the binary bits of the noise image and comparing it with the important image data with the least number of binary bits was implemented. Once the least number of binary bits of the important image data is known usually from the numerical number '1', all images with the summation of binary bit lesser than that will be termed as noise (Figure 9).

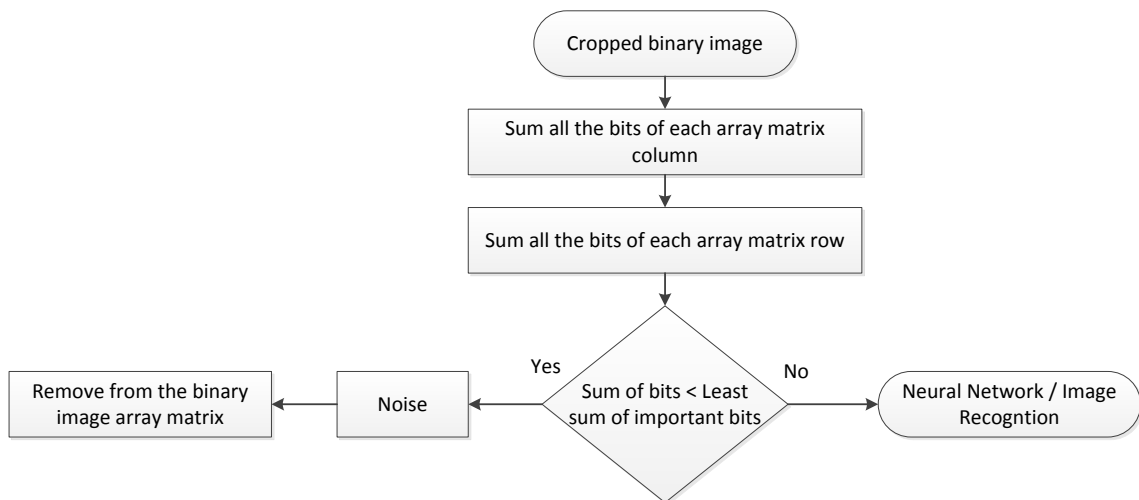


Figure 9: Process Flow Image Noise Reduction

### 3.3.2 SCRATCHES REMOVAL

Scratches removal is almost similar to noise removal from the previous sub-section. However, scratches removal uses the threshold of the highest number of bit of a regular handwritten number to remove invalid handwritten data which have already been scratched out (Figure 10).

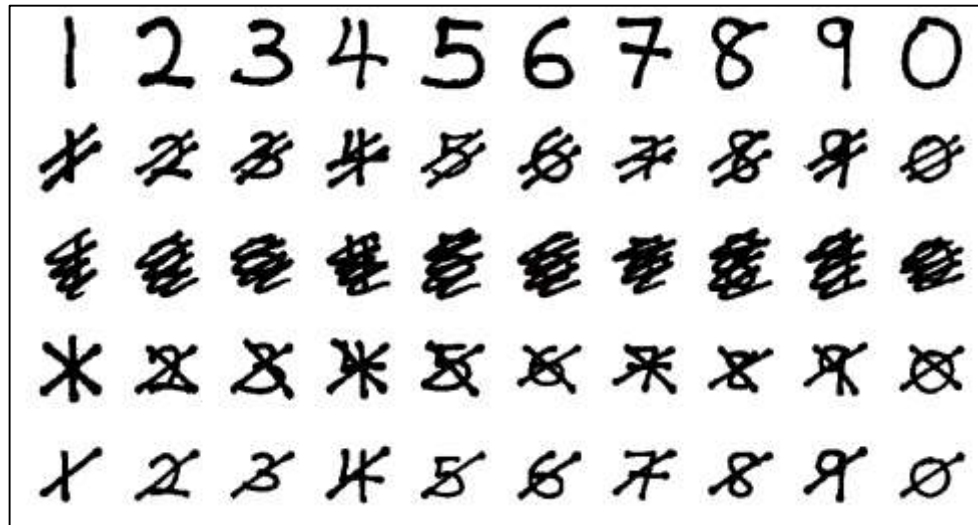


Figure 10: Scratches removal

Several tests have been carried out and it has proven that ‘/’, which is one stroke scratches gives the best results in terms of determining the threshold.

### 3.3.3 SLANT CORRECTION

This phase of image pre-processing is to reduce the variations of handwriting by different humans. There is a tendency that different types of handwritings which are written slanted. However, when the slant is corrected, the handwritten number appears the same as most of the non-slanted handwritten data (Figure 11).



Figure 11: Numbers before and after slant correction

The slant of a handwritten number can be corrected by calculating the angle of slant of the handwritten data and readjust it based on the angle of slant obtained. This is done by considering two points from each side of the handwritten number and obtaining a straight line before calculating its angle.

### **3.4 FEATURE EXTRACTION MODULES**

Table 1 shows three tests were conducted using distinctive feature extraction modules. The first test utilizes only Character Vector Module as its feature extraction module to be used as neural network test samples [11]. The second test was later enhanced with a combination of several feature extraction modules namely the Character Vector Module, Kirsch Edge Detection Module, Alphabet Profile Feature Extraction Module, Modified Character Module and Image Compression Module [1]. With results known that the feature extraction methods used in [1] could not achieve up to 90% time frame accuracy, another feature extraction module was added to the existing modules of Test 3, which is Curvature Vector Module. Each of these modules have its distinct characteristics and has to be known before being trained in the neural network to recognize handwritten data. Test 1 and Test 2 were conducted as control tests and simulation results to validate previous researches with different sets of handwriting collections.

The input that will be used to be fed into the neural network system is the summation of each individual feature extraction matrixes. In test 1, there will be only 35 input elements (Appendix A), while in test 2 there will be 380 input elements (Appendix B). In test 3, there are a total of 388 input elements (Appendix C).

Table 1: Comparison between matrix sizes of two tests

Test 1			Test 2			Test 3		
No	Feature Extraction Module	Matrix Size	No	Feature Extraction Module	Matrix Size	No	Feature Extraction Module	Matrix Size
1	Character Vector Module	5 x 7	1	Character Vector Module	5 x 7	1	Character Vector Module	5 x 7
						2	Kirsch Edge Detection Module	4 x 8 x 8
			2	Kirsch Edge Detection Module	4 x 8 x 8	3	Alphabet Profile Feature Extraction Module	1 x 10
			3	Alphabet Profile Feature Extraction Module	1 x 10	4	Modified Character Module	1 x 15
			4	Modified Character Module	1 x 15	5	Image Compression Module	8 x 8
			5	Image Compression Module	8 x 8	6	Curvature Vector Module	1 x 8
Total size		1 x 35	Total size		1 x 380	Total size		1 x 388

### 3.4.1 CHARACTER VECTOR MODULE

This module is the most basic module in the image processing feature extraction elements. A character vector, with the weight of shaded bits within a region of 5 x 7 matrix is produced and stored in an array matrix (Figure 12).

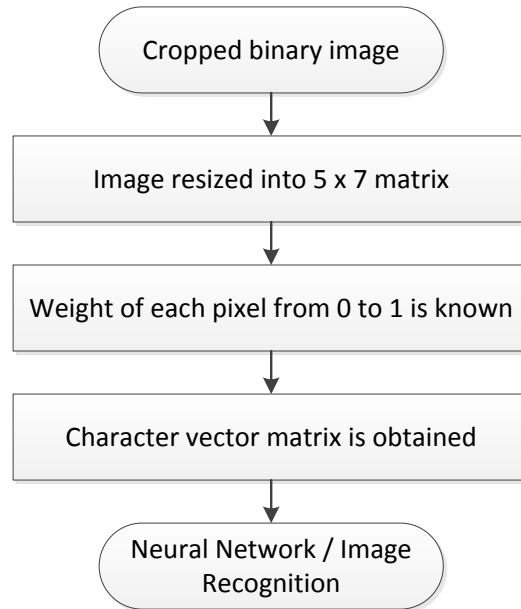


Figure 12: Process Flow of Character Vector Module Algorithm

### 3.4.2 KIRSCH EDGE DETECTION MODULE

This module involves convolution of predefined masks to detect horizontal, vertical, right diagonal and left diagonal edges. The outcome of the module gives the maximum edge strength of each mask in the form of 4 x 8 x 8 matrix, which is then stored in an array matrix (Figure 13).



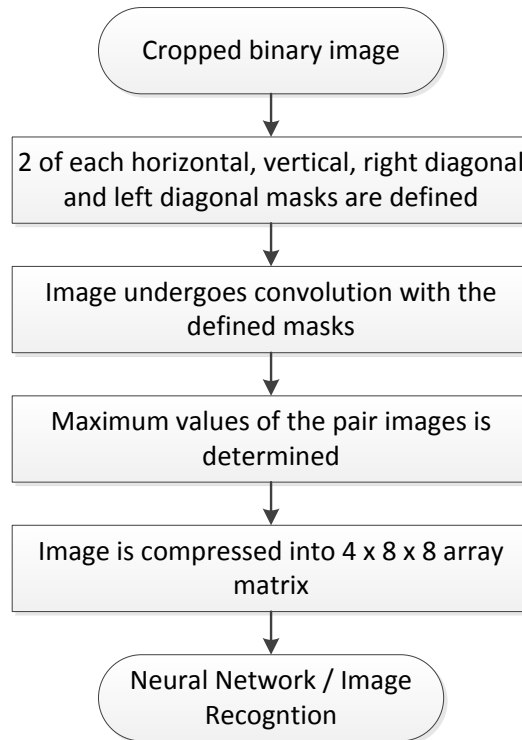


Figure 13: Process Flow of Kirsch Edge Detection Module Algorithm

### 3.4.3 ALPHABET PROFILE EXTRACTION MODULE

This is the module which detects the smoothness of the cropped image numerical handwritten data and also the edge profile by interpreting the image line by line. It also detects the size, in the form of width and height of the cropped image data. As this module highlights only the global feature of the data, the output of the module only generates 1 x 10 floating point matrix (Figure 14).

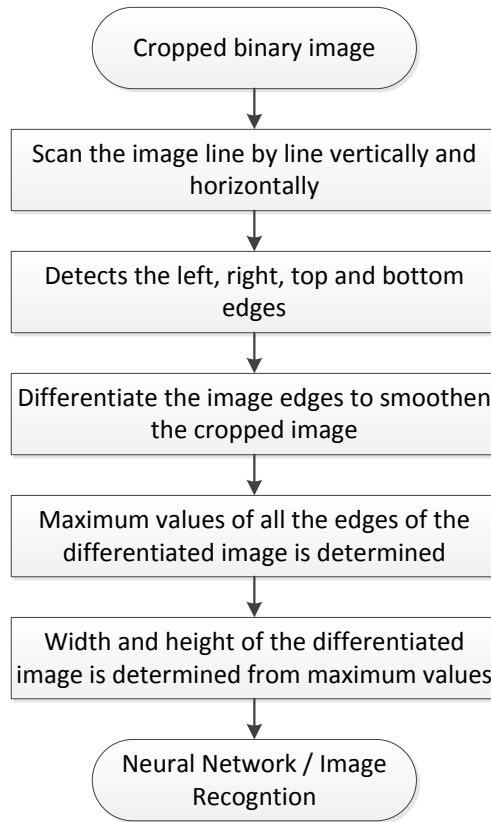


Figure 14: Process Flow of Alphabet Profile Feature Extraction Module Algorithm

### 3.4.4 MODIFIED CHARACTER MODULE

In this module, the input binary image must first be resized into 32 x 32 matrix. Predefined 3 horizontal, 8 vertical and 4 diagonal regions are mapped onto the resized input binary image. The number of bits in the specific region of the data is calculated and divided with the size of the region to obtain an outcome of 1 x 15 floating point matrix. In this method, it is clearly seen that the important feature of each handwritten number is extracted before being fed into neural network or undergo image recognition (Figure 15).

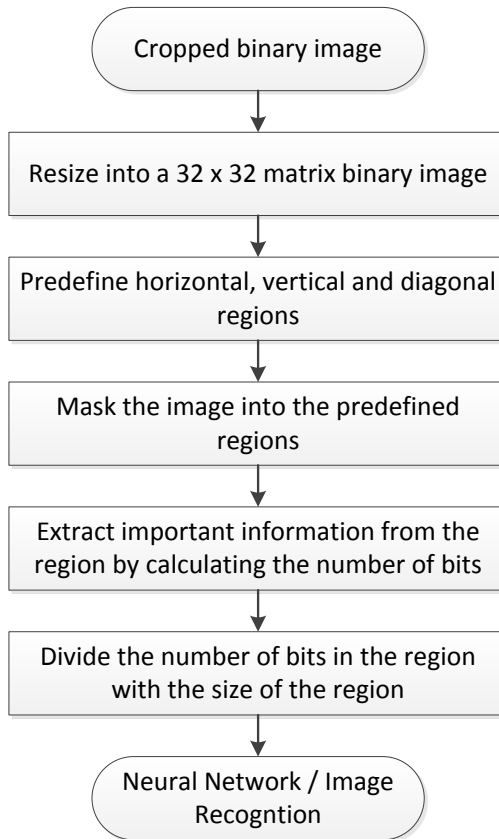


Figure 15: Process Flow of Modified Character Module Algorithm

### 3.4.5 IMAGE COMPRESSION MODULE

This module is used to reduce the size of the neural network input by 16 times so that less processing units is needed to train the neural network or for image recognition. The input of the module will have to be resized into 32 x 32 matrix array, and the output will be compressed into an 8 x 8 floating point matrix (Figure 16).

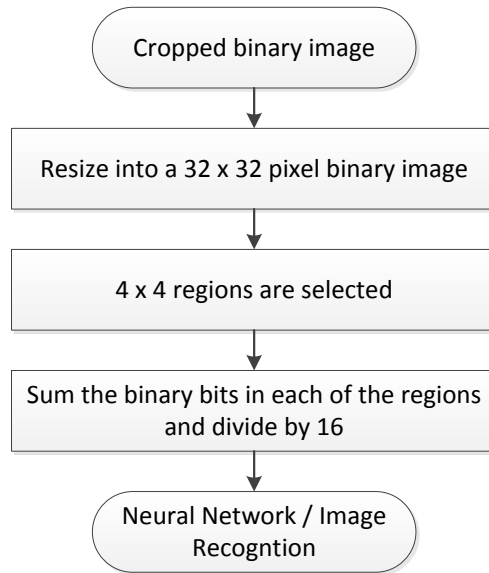


Figure 16: Process Flow of Image Compression Module Algorithm

### 3.4.6 CURVATURE VECTOR MODULE

This module is used to determine the number of bits that forms a curve based on particular regions on the handwritten character. There are 4 regions each on the top half and bottom half of the number which will later form an output of 1 x 8 floating point matrix. This method is important to know which number of the handwritten data has curves and circles which will form a distinctive feature to differentiate the numbers (Figure 17).

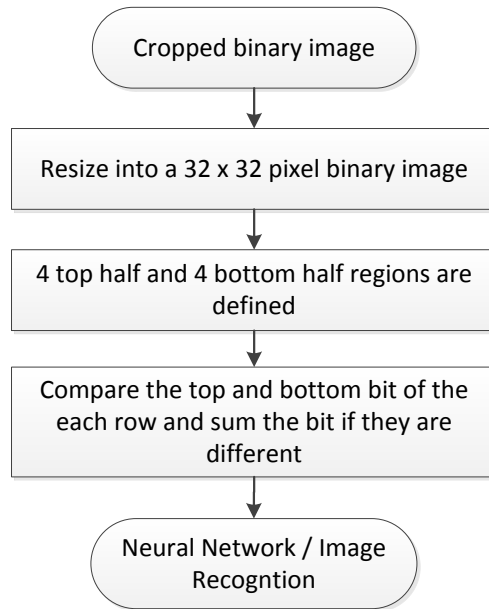


Figure 17: Process Flow of Curvature Vector Module Algorithm

### 3.5 NEURAL NETWORK

The neural network system is the core architecture of this numeric handwritten recognition systems project. Its architecture, including its number of input layer neurons, hidden layer neurons, output layer neurons and training parameters have to be well defined before training begins. The number of input layer neurons is equivalent to the number of input elements extracted from the feature extraction modules. The number of output layer neurons is meanwhile, equivalent to the number of handwritten numerals that is targeted, which is 0 to 9. Thus, there will be 10 output layer neurons. The hidden layers of the neural network system cannot be observed or predicted through the output nor input behavior of the system. Hence, suitable number of neurons in the hidden layer can be obtained heuristically. It is known that complex pattern recognition such as handwriting recognition cannot be trained using little amount of hidden neurons; as the accuracy will be very low. However, if the number of hidden neurons appear to be too large, the system computational burden will increase dramatically and the network might just memorize all the training data input. Therefore, a suitable amount of hidden layer neurons have to be defined, as it is known that the greater amount of hidden neurons present in the layer, the higher the accuracy of the handwritten recognition system capabilities. This is due to the fact that the artificial neural network system has limited capacity in terms of computing large amount of input data. Hence, the more input data and hidden neuron supplied to the system will actually complicate and confuse the network to generate the function to compute the output. In order to achieve better recognition results and the need to balance computational load, there are certain rules to select the amount of hidden neurons to be used to predict the output based on all the input data; which is that the amount of hidden neurons must be between the number of input neurons and output neurons, as well as choosing the best parameters which gives the lowest validation error.

Table 2 shows the training parameters used to train the Multilayer Feed Forward Back Propagation Neural Network systems. These are the standardize parameters to obtain the optimum network performance for numeric handwriting recognition system. In test 1, the system is trained with 5 sets of handwritten numerals (Appendix D), while in test 2,

the system is trained with 30 sets of handwritten numerals (Appendix E). However, both test 1 and test 2 are only used as a control set and verification of previous work. In test 3 (Appendix F), a finalized neural network system is trained with 15 sets of handwritten numerals.

Table 2: Neural Network Training Parameters

Training Parameters	Definition
Performance function = Sum squared Error	Total of squared errors from the training predictions
Goal = 0.01	Minimum error achieved before the training stops
Epochs = 5000	Maximum number of iterations before training stops
Momentum = 0.95	Fraction of weight among the neurons in the network layers
Activation Function = Logsig	A transfer function to calculate the output based on its input values
Training Functions = traingdx	Updates weights and biases during training according to momentum values

### 3.6 IMAGE ACQUISITION MODULE

This Handwriting Recognition using Webcam for Data Entry project is designed to use two web cameras to capture the Region of Interest data on the Universiti Teknologi PETRONAS examination score sheet paper, which includes Table Number, Examination Index Number, Students' filled Question No, Examiners' filled Question No, Marks and Total Marks.

MATLAB has its own Webcam Image Acquisition Toolbox which allows users to acquire images from a webcam using its built in webcam functions. The above toolbox have provided an adequate platform for the webcams to be interfaced with MATLAB software, hence, users are able to capture the image and do image processing via MATLAB itself.

Two cameras were used to capture different Region of Interest (ROI), with the first focused on the Table Number and Examination Index Number, as shown in Figure 18, while the second being focused on the Students' filled Question No, Examiners' filled Question No, Marks and Total Marks, as shown in Figure 19.

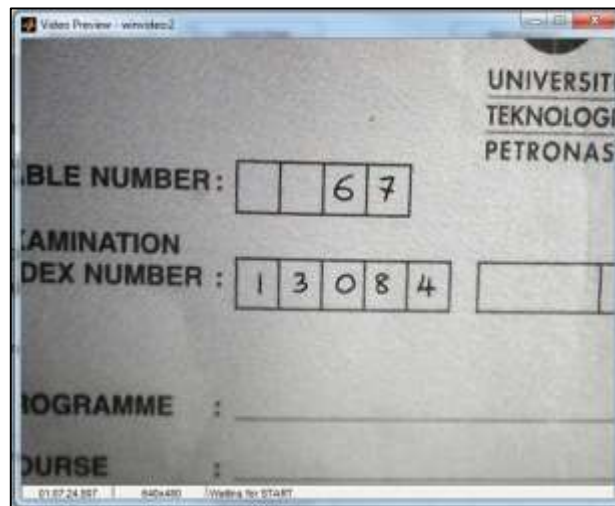


Figure 18: Output from first camera



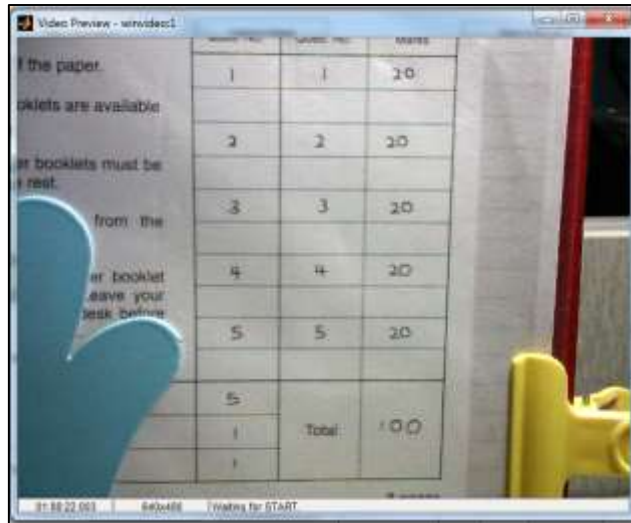


Figure 19: Output from second camera

### 3.6.1 PROTOTYPE

A simple prototype was built to simple capture the two important regions of the examination score sheet as shown in Figure 20. The two cameras which was used were 12 Megapixel Driverless Night Vision Webcam PC USB CMOS Camera, which however only gives approximately 0.3 Megapixel video recording images. Since the camera module is driverless, it is compatible with MATLAB software and can be easily accessible through its internal built-in functions.

The 12 Megapixel Driverless Night Vision Webcam PC USB CMOS Camera has the following specifications as stated by its manufacturer:

- Default resolution: 640 x 480 = 307200 pixel
- Focus range: Manual focus from 3cm
- White balance: Auto
- Exposure control: Auto
- Interface: USB 1.1/2.0
- Working temperature: 0 °C to 40 °C

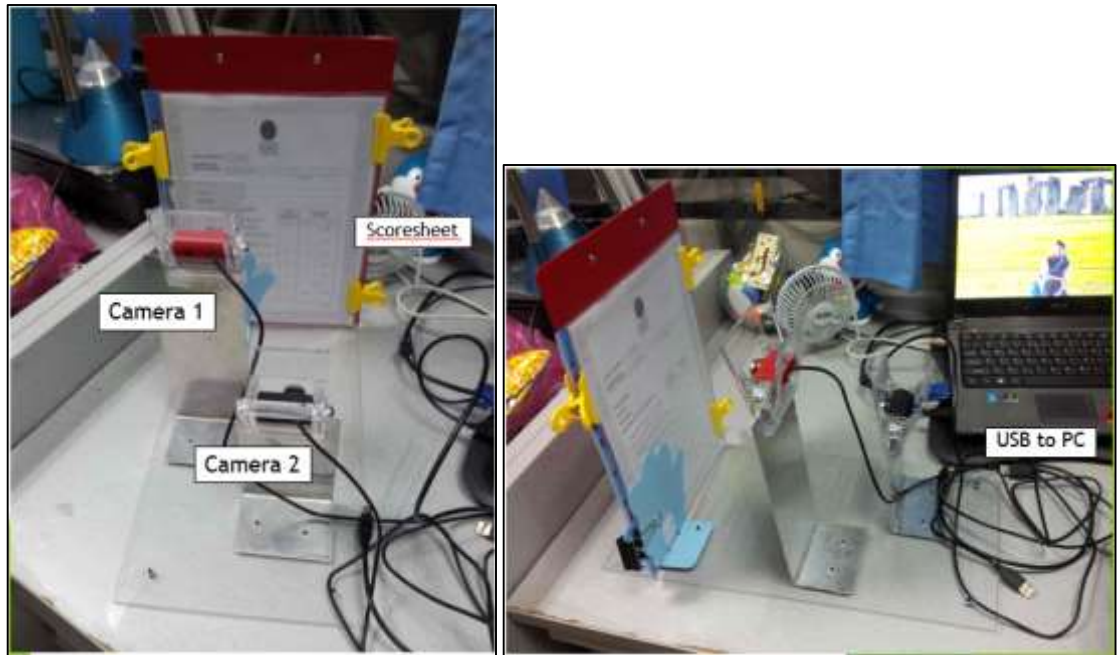


Figure 20: Location of the camera and the prototype

However, there are limitations of using this camera as the default pixel count is only about 0.3 Megapixel. Thus, the image captured by the camera might not have enough number of bits or pixels to be pre-process and fed into the artificial neural network system. This is also the reason why two cameras were being built onto the prototype instead of one, as one camera could not capture the whole image and still give a good amount of pixels for the recognition system to process its data.

### 3.6.2 ACQUIRING THE REGION OF INTEREST

The region of interest of each number is predefined as shown in Figure 21, whereby each box is cropped by adjusting manually the vertical and horizontal axis values. Once each number is being cropped, it will undergo pre-processing and feature extraction modules, before being fed onto the neural network for recognition purposes.

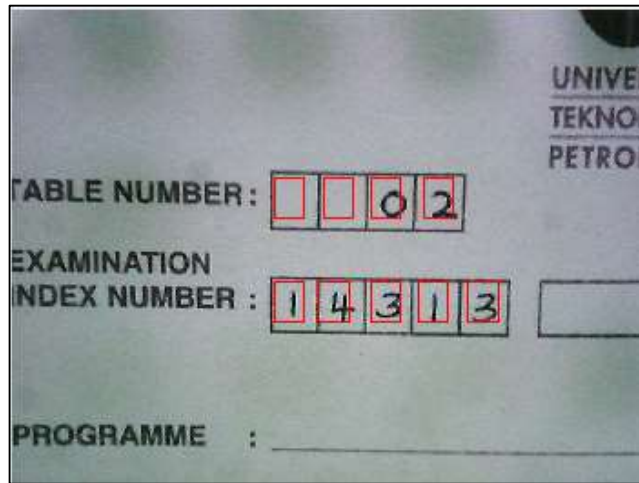


Figure 21: Defining the region of interest

### 3.7 GRAPHICAL USER INTERFACE MODULE

Graphical User Interface (GUI) is a user interactive system which enhances the usability and purpose of the program. It allows users to get access and key in data, as well as obtaining the output from its layout. In short, the GUI system is a system that will be used by a user to determine the appropriate function or command to be called to execute the output. A GUI layout was designed as shown in Figure 22.

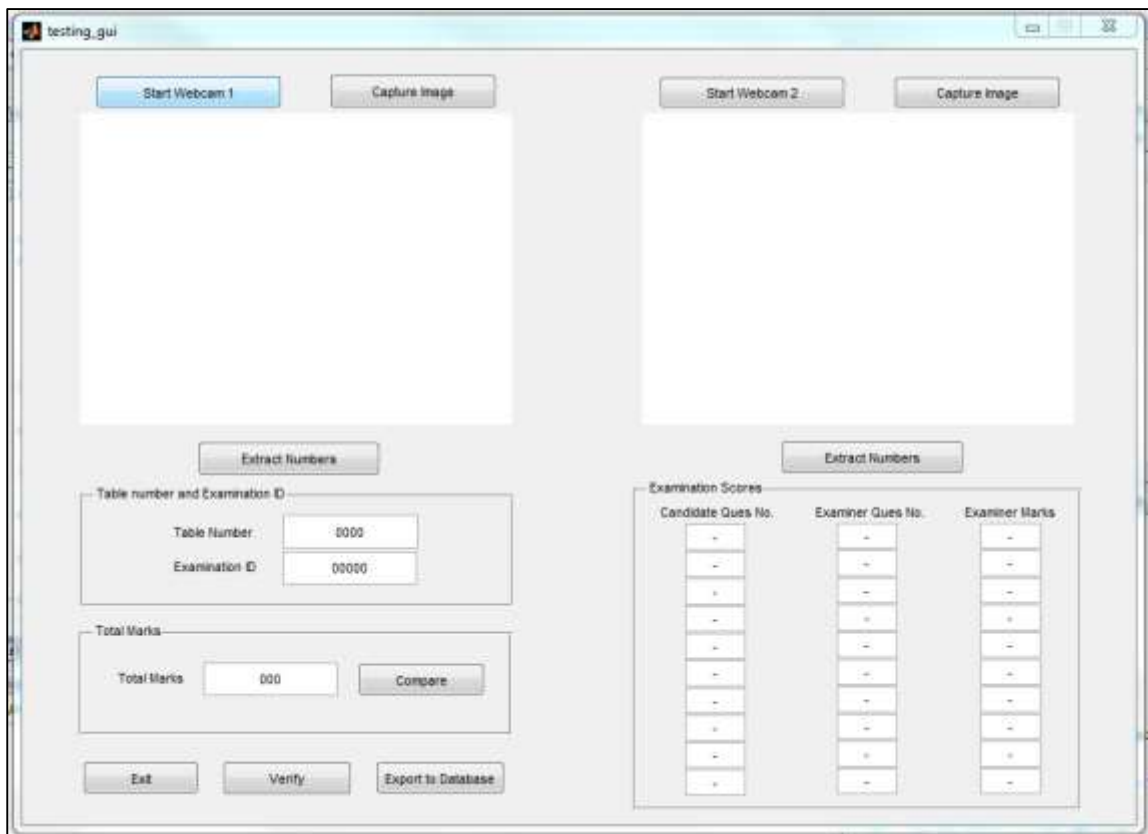


Figure 22: Graphical User Interface Layout for Handwriting Recognition System using Webcam for Data Entry

### 3.7.1 GRAPHICAL USER INTERFACE COMMAND STEPS

The following command steps, as shown in Figure 23 were designed to execute the output.

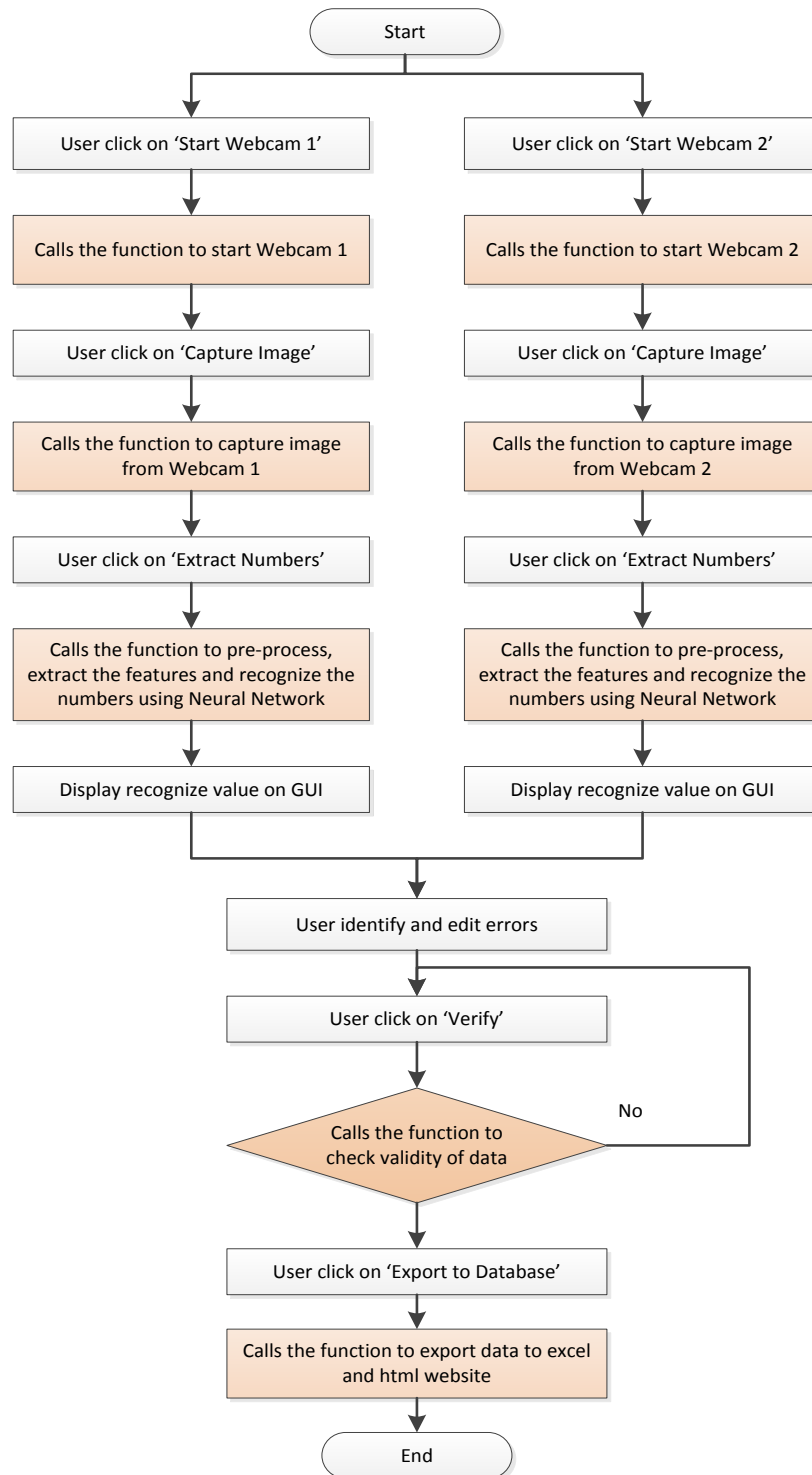


Figure 23: GUI Command Steps

In order for the current data entry system to function, the user must follow the following steps stated on the GUI Command Steps. The user first clicks ‘Start Webcam 1’ and the video preview from the first webcam is shown. Next, the user clicks on ‘Capture Image’ in which the current video preview image from Webcam 1 is taken and displayed on the GUI. When the user clicks on ‘Extract Numbers’, pre-processing, feature extraction and feeding the image onto the trained neural network is being carried out, and the output will be displayed on the Table number and Examination ID column. The user can recalibrate the data if necessary (Figure 24).

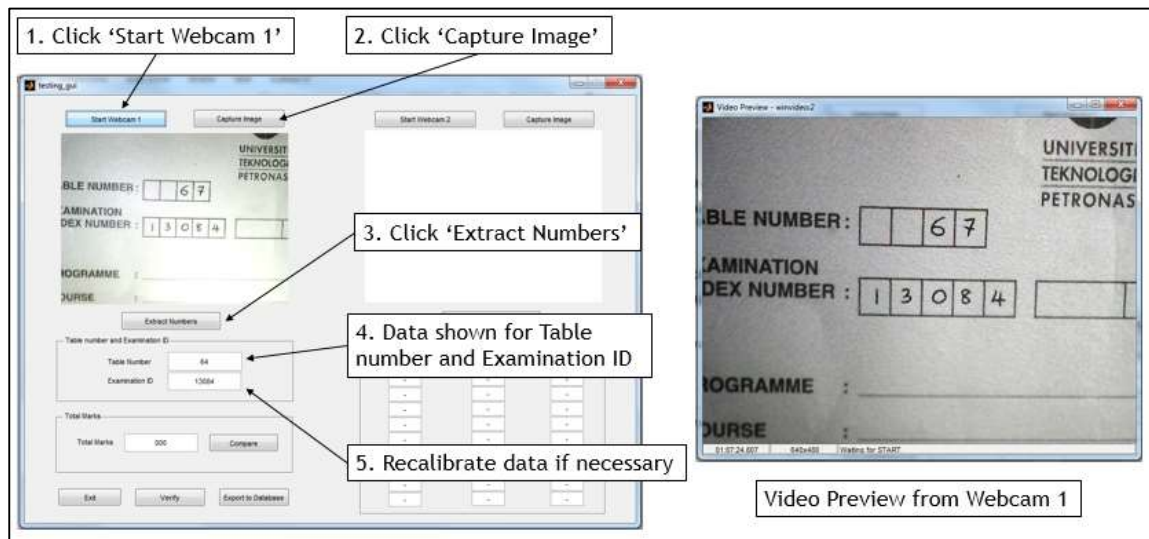


Figure 24: Extracting data from Webcam 1 onto the GUI

Following that, the same process is being carried out for Webcam 2, in which the user clicks ‘Start Webcam 2’ and the video preview from the second webcam is shown. Next, the user clicks on ‘Capture Image’ in which the current video preview image from Webcam 2 is taken and displayed on the GUI. When the user clicks on ‘Extract Numbers’, pre-processing, feature extraction and feeding the image onto the trained neural network is being carried out, and the output will be displayed on the Candidate Ques No., Examiner Ques No., Examiner Marks and Total Marks column. The user can again recalibrate the data if necessary (Figure 25).

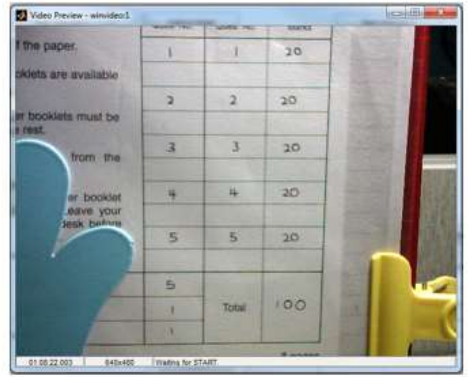
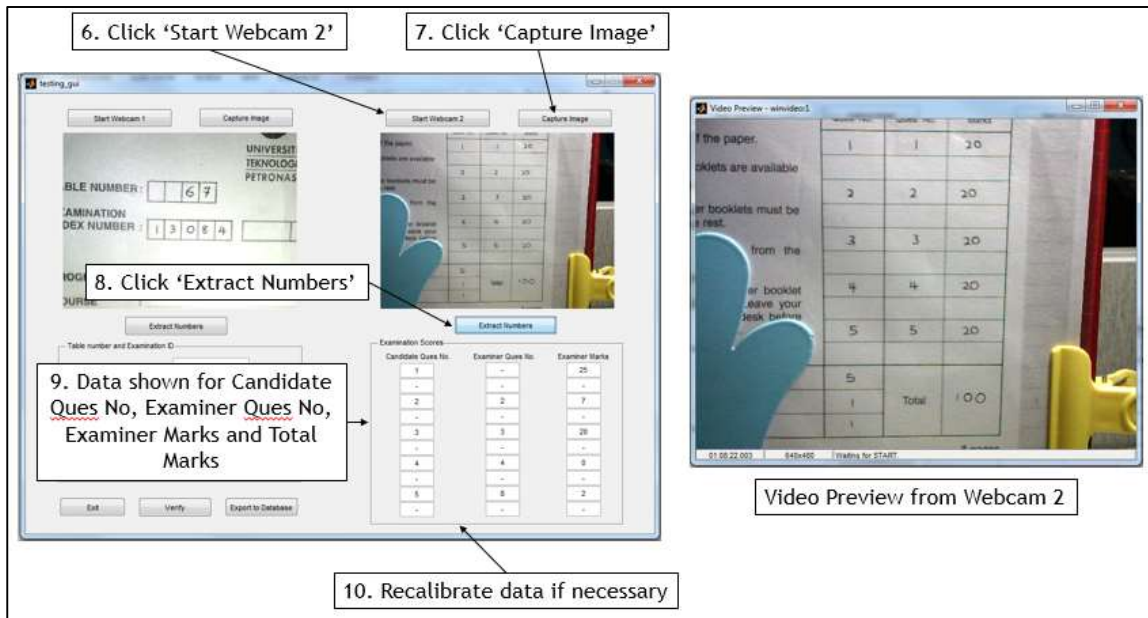


Figure 25: Extracting data from Webcam 2 onto the GUI

Once extracting the data from the Webcams is complete, user could click on the 'Compare' button on the GUI, which calculates the Total Examiner Marks and match the data with the total marks recorded (Figure 26).

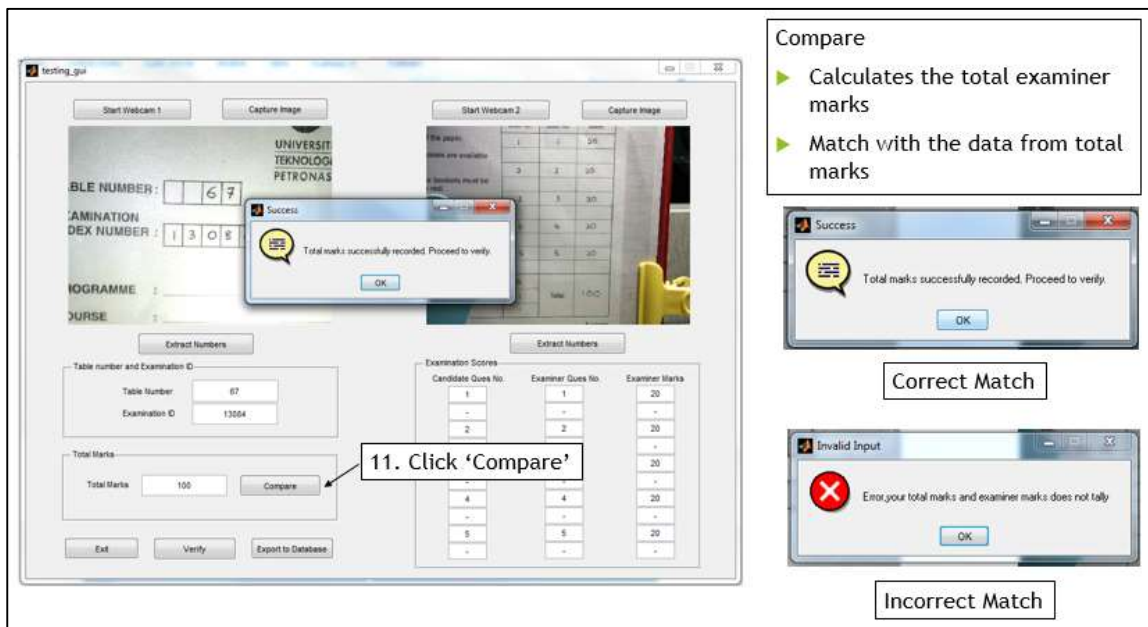


Figure 26: Comparing the total marks with sub marks on GUI

Subsequently, the user have to click on the ‘Verify’ button to ensure all the data extracted are correct and ready to be exported onto the database. Since the scope is focused on UTP examination score sheet, it is typical that examiner marks does not exceed 20 marks and total question numbers does not exceed 5. Hence, this gives flexibility to the students and examiners to write data on the score section of the score sheet. However, this system function can be altered accordingly to give more flexibility and freedom to the user depending on the number of questions and total marks for each question in the particular examination course. The ‘Verify’ button also ensures that the user does not key in more than 5 inputs and the data of question number and marks should be on the same row. Since there are two examiner and student question number which could be valid, the examiner question number is taken priority onto the database. However, if the examiner question number column is empty, the student question number is taken into priority to be keyed onto the database (Figure 27).

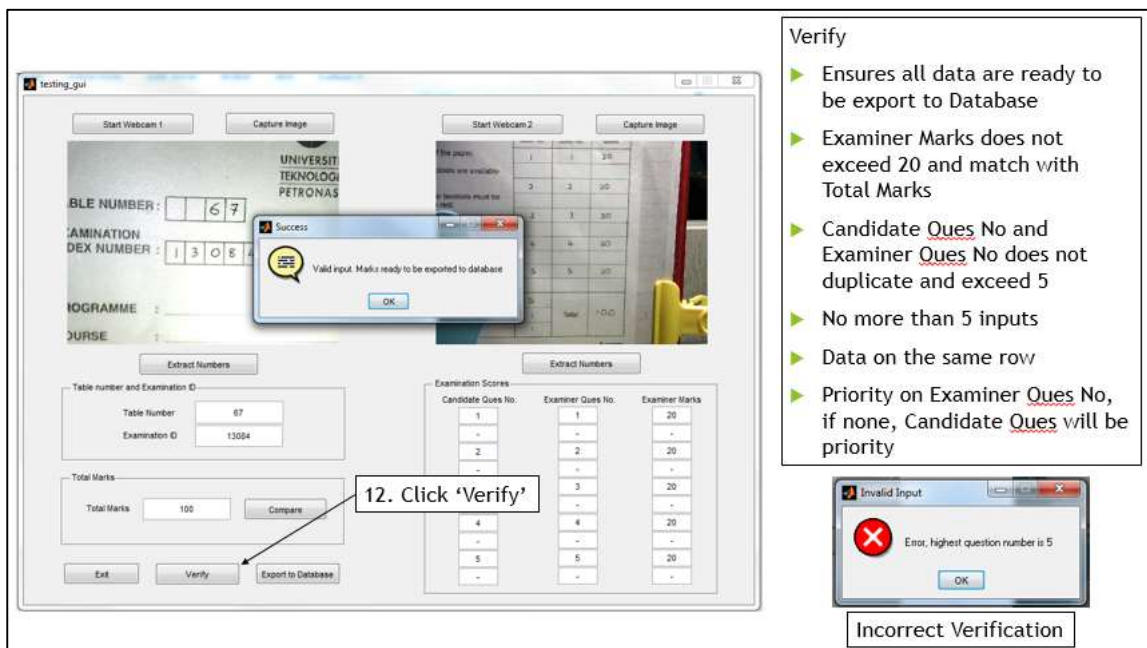


Figure 27: Verifying data entry on GUI

The following Figure 28, shows a completed step of the executed Graphical User Interface on recognizing handwritten data from the UTP examination score sheet. Once the ‘Export



to Database' button is clicked by the user. The system will rearrange and gather the appropriate data to be transferred to the Excel Database (Figure 29) and HTML website (Figure 30).

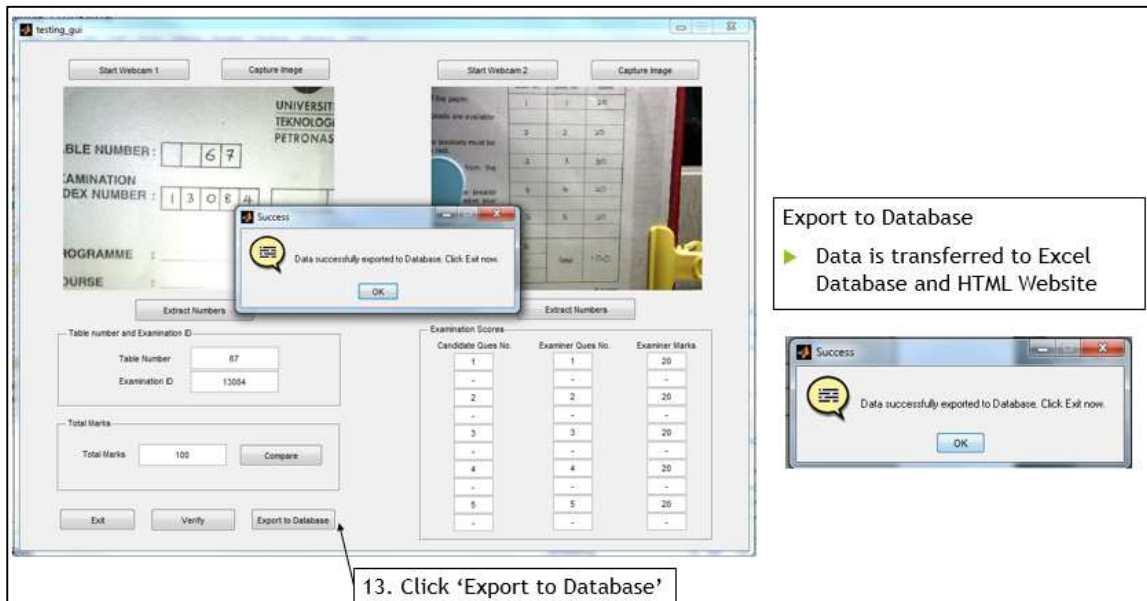


Figure 28: Completed GUI

As for the data entry system, it doesn't matter how the data is arranged as the system is able to rearrange the data, even if it is not in order. If any of the extracted data is empty, the outcome of will be recorded as 0.


	A	B	C	D	E	F	G	H
1	 <b>UNIVERSITI TEKNOLOGI PETRONAS</b> <i>engineering futures...</i>							
2								
3								
4								
5	<b>Handwritten Recognition System for Data Entry</b>							
6	Question Numbers							
7	<b>Table number</b>	<b>Examination ID</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>Total Marks</b>
8	67	13084	20	20	20	20	20	100
9	123	34018	17	14	3	20	15	69
10	93	21005	10	18	19	0	0	47
11	32	15291	15	0	12	18	0	45
12	71	14941	0	1	0	1	0	2
13	1230	1	10	20	10	0	0	40
14	19	14312	18	10	20	3	6	57
15	32	15980	7	12	3	10	8	40
16	1	12369	2	7	0	16	0	25
17								

Figure 29: Exported data to the Excel Database


<b>Handwriting Recognition System for Data Entry Database</b>							
 <b>UNIVERSITI TEKNOLOGI PETRONAS</b> <i>engineering futures...</i>							
<b>Handwritten Recognition System for Data Entry</b>							
Question Numbers							
<b>Table number</b>	<b>Examination ID</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>Total Marks</b>
67	13084	20	20	20	20	20	100
123	34018	17	14	3	20	15	69
93	21005	10	18	19	0	0	47
32	15291	15	0	12	18	0	45
71	14941	0	1	0	1	0	2
1230	1	10	20	10	0	0	40
19	14312	18	10	20	3	6	57
32	15980	7	12	3	10	8	40
1	12369	2	7	0	16	0	25

Figure 30: Exported data to the Excel Database

### 3.8 GANTT CHART & KEY MILESTONES

The following tables, Table 3 and Table 4 depict a detailed timeline of the work involved within the 14 weeks of FYP I and FYP II respectively.

Table 3: Gantt Chart and Key Milestones for FYP I

Details/Week	FYP I													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Literature Review	■	■	■	■	■									
Extended Proposal						●								
System Identification & Modeling						■	■	■						
Proposal Defense								●						
Development of algorithm									■	■	■	■	■	■
Interim Report														●

● Key milestone

■ Process

Table 4: Gantt Chart and Key Milestones for FYP II

Details/Week	FYP II													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Development of algorithm	■	■	■	■	■									
Samples gathering, simulation and implementations				■	■	■	■	■						
Progress report							●							
Comparative analysis							■	■	■	■				
Pre-SEDEX											●			
Draft final report												●		
Dissertation													●	
Technical paper													●	
Project viva														●

● Key milestone

■ Process

## CHAPTER 4: RESULTS AND DISCUSSION

### 4.1 HANDWRITING RECOGNITION TESTS

#### 4.1.1 TEST 1

Test 1 Handwriting Recognition System only includes Character Vector Module as its feature extraction module. Hence, the system is being tested with 2 types of samples (Appendix G):

- Sample 1: The training set
- Sample 2: A test set by 3 random people who wrote 5 sets each

Both the sample sets are scanned using a scanner and ran into the system to determine its accuracy. All the important information from Test 1, such as mistakes, occurrences and percentage of accuracy are tabulated with each handwritten number data in Table 5.

Table 5: Percentage of accuracy of Test 1

Type of samples		Sample 1		Sample 2	
Handwritten Numbers	Occurrences	Mistakes	Percentage of Accuracy	Mistakes	Percentage of Accuracy
1	15	0	100%	6	60.00%
2	15	0	100%	8	46.67%
3	15	0	100%	13	13.33%
4	15	0	100%	7	53.33%
5	15	0	100%	1	93.33%
6	15	0	100%	1	93.33%
7	15	0	100%	3	80.00%
8	15	0	100%	10	33.33%
9	15	0	100%	4	73.33%
0	15	0	100%	0	100.00%
Total	150	Average	100%	Average	64.67%

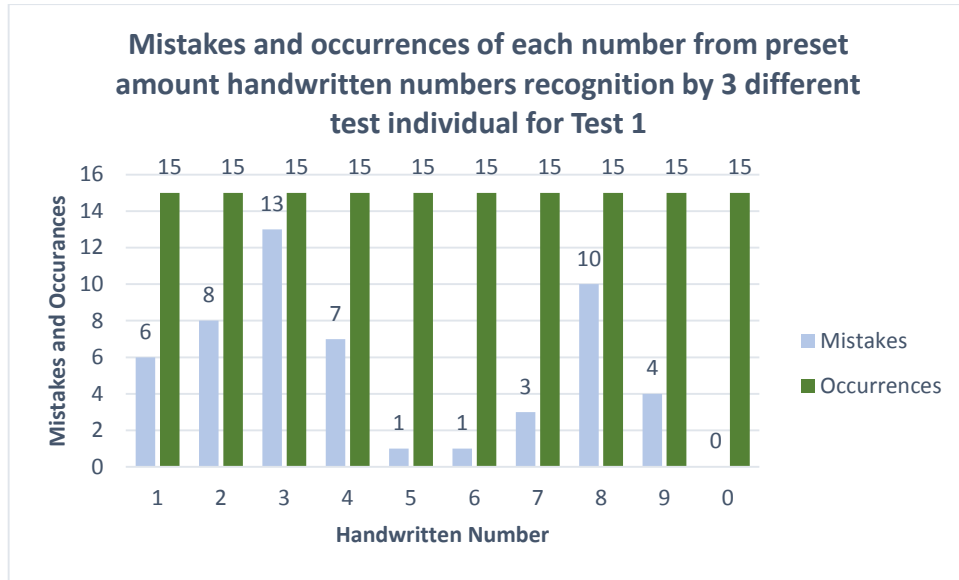


Figure 31: Mistakes and occurrences of each number in Test 1

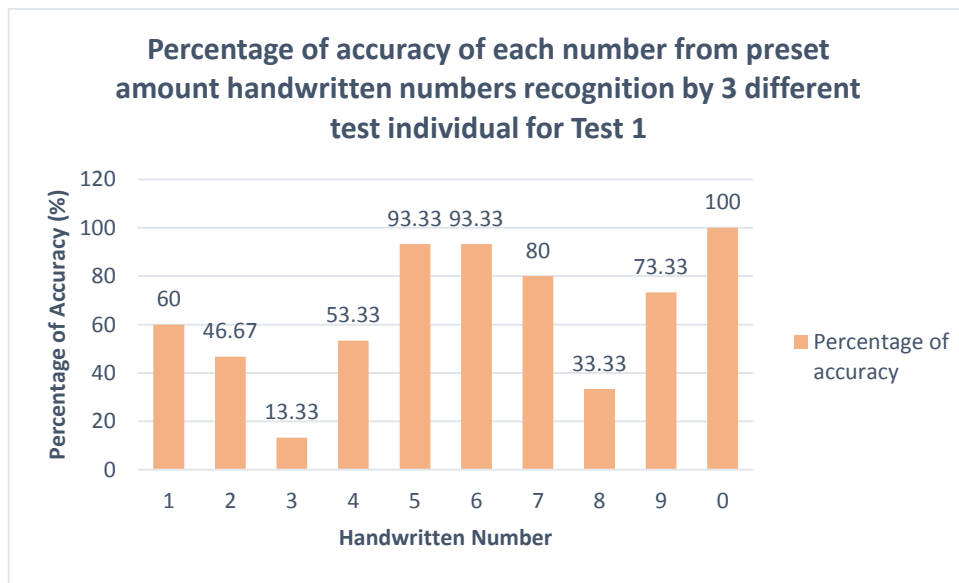


Figure 32: Percentage of accuracy of each number in Test 1

Based on Table 5, it is known that handwriting recognition using neural network has no issues recognizing and distinguishing the numbers in the original training set. However, the results shows reduction in accuracy when tested with a test set which has not been analyzed before by the neural network. From 100% accuracy in Sample 1, the results plunged to an average of 64.67% accuracy.

Figure 31 and Figure 32, shows the information of mistakes, occurrences and percentage of accuracy of each number tested using the neural network. In this Test 1, the number with the highest failure rate seems to be '3' followed by '8'. Other numbers such as '1', '2', '4', '7' and '9' have unacceptable percentage of accuracy as well. To clarify this situation, it can be seen that the Character Vector feature extraction alone is not able to extract the best features from each region of the handwritten number. The numbers '3' and '8' appears to have high failure rate because both these numbers can be written at a very similar way whereby, most of the regions of the feature extraction module might have similar or close enough number of bits especially the right side region, which the neural network system could not distinguish. The other numbers as mentioned which are, '1', '2', '4', '7' and '9' could possibly be facing the same issue as some of the regions might have similar number of bits in it, causing confusion to the trained neural network system. From this test, it is concluded that training a neural network using only a simple Feature Extraction method, which is Character Vector Module is insufficient to produce a robust handwriting recognition system for data entry. The system is further improved in Test 2.

#### **4.1.2 TEST 2**

Test 2 Handwriting Recognition System includes the combination of several Feature Extraction method to enhance the recognition accuracy, namely Character Vector Module, Kirsch Edge Detection Module, Alphabet Profile Feature Extraction Module, Modified Character Module and Image Compression Module. Since we have known that the system has no issues identifying handwritten data in the training set, the system is being tested with 2 different types of samples (Appendix H):

- Sample 3: A test set by 15 random people who wrote 1 set each
- Sample 4: A test set by 1 random people who wrote 15 sets, but with random arrangement of numbers

Both the sample sets are again scanned using a scanner and ran into the system to determine its accuracy. All the important information from Test 2, such as mistakes, occurrences and percentage of accuracy are tabulated with each handwritten number data in Table 6.

Table 6: Percentage of accuracy of Test 2

Type of samples	Sample 3			Sample 4				
Handwritten Numbers	Occurrences	Mistakes	Percentage of Accuracy	Occurrences	Mistakes	Percentage of Accuracy	Average	
1	15	2	86.67%	26	7	73.08%	79.88%	
2	15	3	80.00%	15	0	100.00%	90.00%	
3	15	3	80.00%	17	0	100.00%	90.00%	
4	15	2	86.67%	10	0	100.00%	93.34%	
5	15	7	53.33%	9	7	22.22%	37.78%	
6	15	1	93.33%	12	1	91.67%	92.50%	
7	15	0	100%	12	2	83.33%	91.67%	
8	15	5	66.67%	12	0	100.00%	83.34%	
9	15	2	86.67%	19	0	100.00%	93.34%	
0	15	2	86.67%	18	0	100.00%	93.34%	
Total	150	Average	82%	150	Average	87.03%		
		Average						84.52%

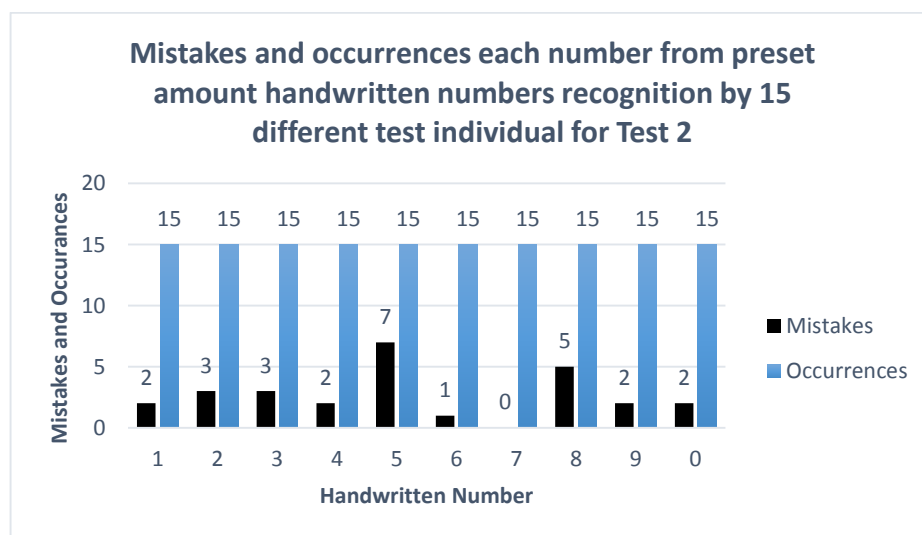


Figure 33: Mistakes and occurrences of each number in Sample 3 of Test 2



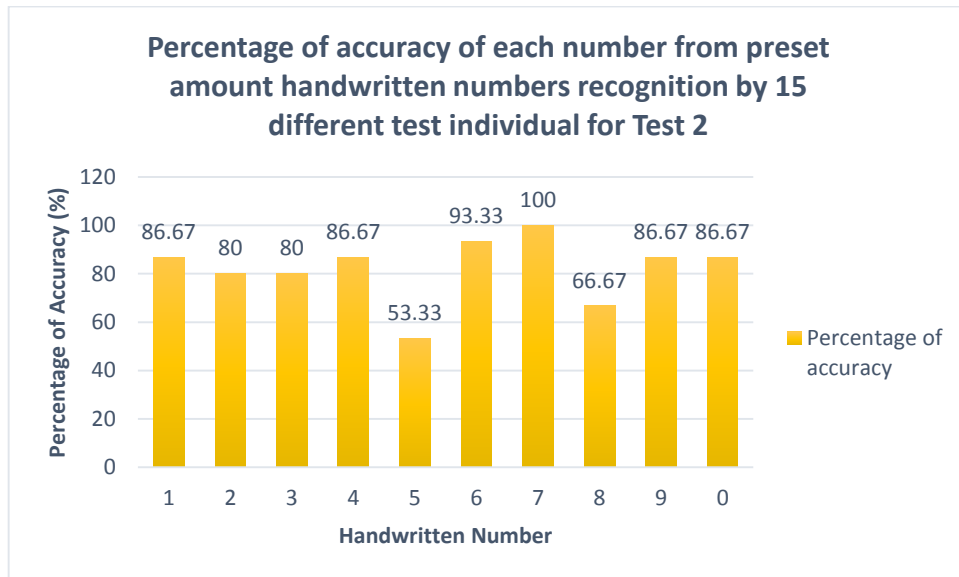


Figure 34: Percentage of accuracy of each number in Sample 3 of Test 2

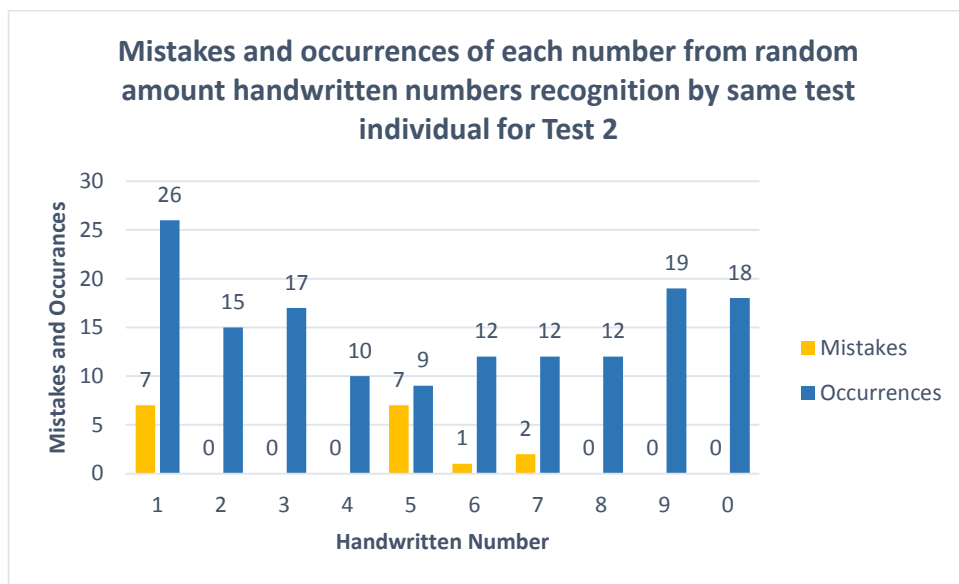


Figure 35: Mistakes and occurrences of each number in Sample 4 of Test 2

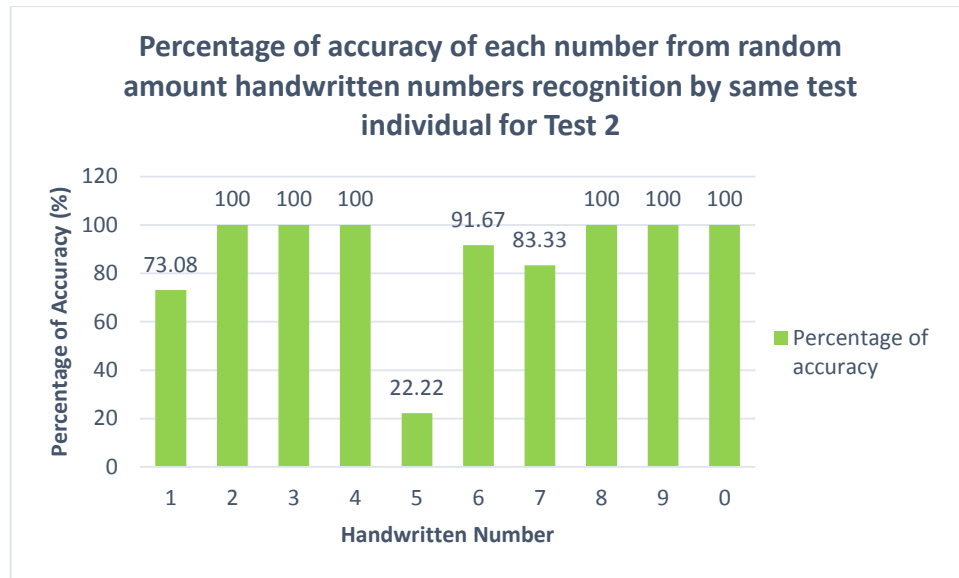


Figure 36: Percentage of accuracy of each number in Sample 4 of Test 2

Based on Table 6, it is known that handwriting recognition using neural network with multiple feature extraction modules of Character Vector Module, Kirsch Edge Detection Module, Alphabet Profile Feature Extraction Module, Modified Character Module and Image Compression Module have increased the accuracy rate of recognition. This shows improvements to the recognition system as the test sets are all new and yet to be analyzed before by the neural network. In this Test 2, the percentage of accuracy among the 2 samples are 82% and 87.04% respectively.

Figure 33 and Figure 34, shows the information of mistakes, occurrences and percentage of accuracy of each number written by 15 random people who wrote 1 set each tested using the new neural network. In this sample 3 of Test 2, the number with the highest failure rate seems to be ‘5’ followed by ‘8’. Other numbers have relatively good recognition accuracy, but can be improved further. Figure 35 and Figure 36, shows the information of mistakes, occurrences and percentage of accuracy of each number written by 1 random people who wrote 15 sets, but with random arrangement of numbers tested using the new neural network. In this sample 4 of Test 2, the number with the highest failure rate still seems to be ‘5’, but followed by ‘1’. Other numbers have again relatively

good recognition accuracy, but can be improved further. These feature extraction methods still lacks the ability to distinguish curved features as appeared in both number ‘5’ and ‘8’.

### 4.1.3 TEST 3

As of Test 3 Handwriting Recognition System, a combination of 6 Feature Extraction methods were used to improve the accuracy of the handwriting recognition system, namely Character Vector Module, Kirsch Edge Detection Module, Alphabet Profile Feature Extraction Module, Modified Character Module, Image Compression Module and last but not least, Curvature Vector Module. Other than that, Test 3 Handwriting Recognition Systems also consist of an additional pre-processing method in which Test 1 and Test 2 does not undergo, which is slant correction, in order to reduce the variations in handwriting of the same number.

In this test, a total of 60 sets of training data were obtained from random people and were used to train the neural network. 8 cases of neural network testing were conducted with each having different training sizes, hidden neurons and training parameters and results were tabulated in Table 7 (Appendix I).

Table 7: Neural Network Training Accuracy

Case	Training Size	Hidden Neurons	Minimum error	Best Results (%)		Avg. Accuracy (%)
				Trained Data	Untrained Data	
1	5 sets	30	0.01	83.33	86.33	84.83
2	10 sets	25	0.001	78.33	83.00	80.67
3	15 sets	15	0.00001	91.00	90.33	90.67
4	20 sets	15	0.00001	66.67	74.00	70.34
5	30 sets	10	0.0001	65.33	72.33	68.83
6	40 sets	10	0.0001	55.33	51.67	53.50
7	50 sets	10	0.0001	35.33	37.67	36.50
8	60 sets	10	0.0001	10.00	10.00	10.00

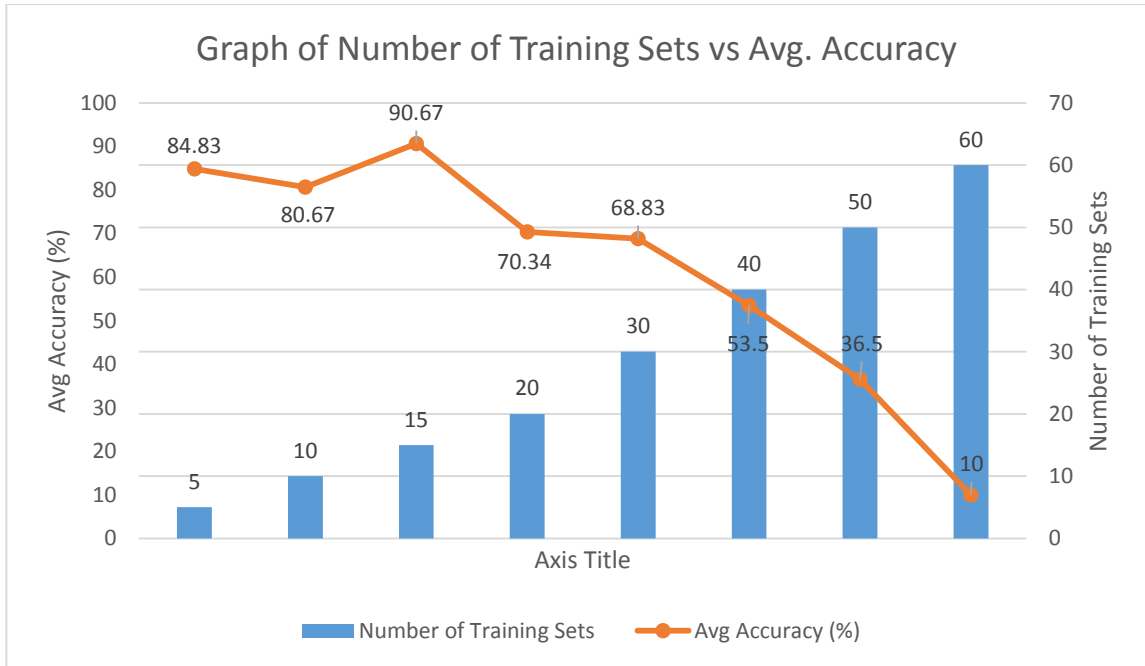


Figure 37: Graph of Number of Training Sets vs Avg. Accuracy

From the Table 7 above, it is known that the accuracy of the training sets peaked at case 3, in which later, the accuracy starts to decline. From Figure 37, it can be deduced that the relationship between training parameters and training sizes were known. In neural network system architecture, it is proven that the increment of various training sets will actually decrease the accuracy of the recognition output. This phenomenon is being rectified here in this test and for the time being, the time-frame target of achieving 90% accuracy on handwriting recognition is achieved, which is by using 15 sets of handwritten training data.

To achieved the accuracy as stated in Table 7, all the tested neural network systems were tested with 30 sets of controlled data from trained data and untrained data each. As stated below:

- Sample 5: Trained data represents the data, in which the handwriting is familiar with the training data and was written by the same person who wrote the training data

- Sample 6: Untrained data represents the data, in which the handwriting is not familiar with the training data and that the training data was written by other random people.

The major problem encountered when training neural network is that, most of the training stops abruptly before the epochs or goals is achieved. Multiple times of training is being carried out again to obtain the highest achievable accuracy before the particular set of neural network system is used to be tested onto the controlled set of testing data. A considerably large amount of time was spend on training this data and by far, the highest accuracy achieved was by using case 3, which the stated parameters and yielded 90.67% of handwriting accuracy (Table 8).

Table 8: Percentage of accuracy of Test 3

Type of samples	Sample 5 Trained Data			Sample 6 Untrained Data			Average	
	Occurrences	Mistakes	Percentage of Accuracy	Occurrences	Mistakes	Percentage of Accuracy		
1	30	0	100%	30	1	96.67%	98.34%	
2	30	5	83.33%	30	2	93.33%	88.33%	
3	30	2	93.33%	30	2	93.33%	93.33%	
4	30	0	100%	30	1	96.67%	98.34%	
5	30	7	76.67%	30	7	76.67%	76.67%	
6	30	1	96.67%	30	3	90%	93.34%	
7	30	9	70%	30	8	73.33%	71.67%	
8	30	1	96.67%	30	2	93.33%	95%	
9	30	1	96.67%	30	2	93.33%	95%	
0	30	1	96.67%	30	2	93.33%	95%	
Total	300	Average	91%	300	Average	90.33%		
		Average						90.67%

#### 4.1.4 COMPARISON AMONG ALL THREE TESTS

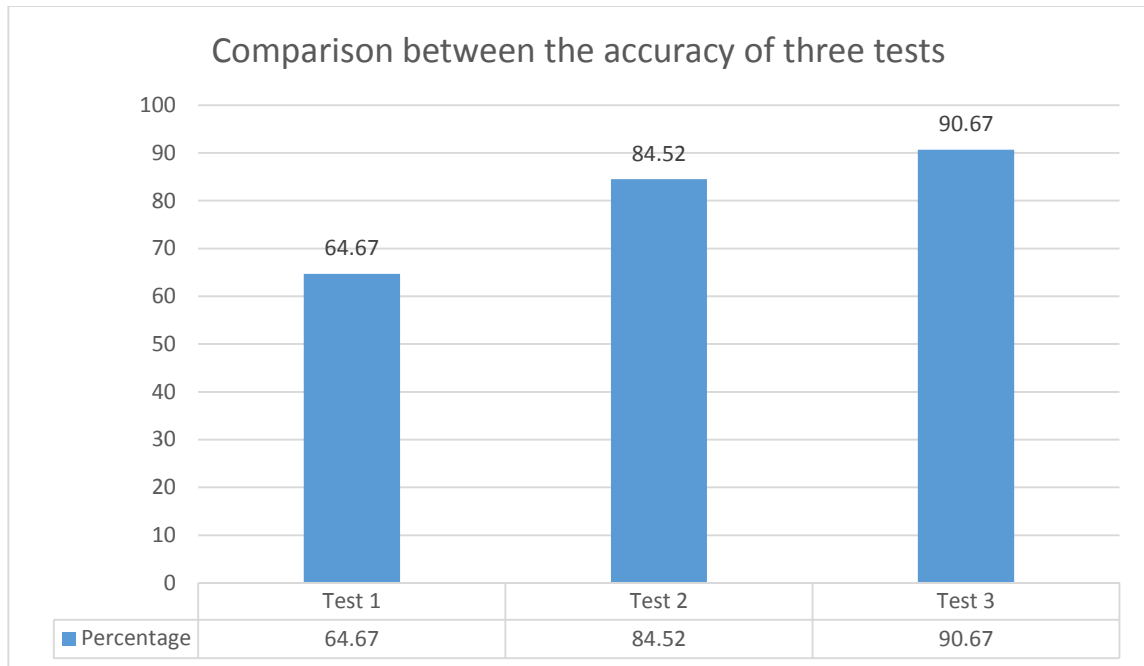


Figure 38: Comparison between the percentages of accuracy between 3 tests

From this test, it is concluded that the newly trained neural network have greatly improved the percentage of accuracy in recognizing handwritten numerical numbers, through increasing the number of feature extraction modules, by 19.85% for Test 2 and 26.00% for Test 3 (Figure 38). As the time-frame target of this project of 90% is achieved, the recognition rate can further be improved, but has to be stopped due to time constraint of the timeline of final year project. The trained neural network was used to implement a full data entry system with GUI (Graphical User Interface).

#### 4.1.5 COMPARISON AMONGST FEATURE EXTRACTION MODULES

For each feature extraction module, a basic neural network is trained and the accuracy is tested to determine the degree of discrimination among different classes of feature extraction modules. From this test, it is known that which feature extraction module gives the highest possible accuracy to distinguish one digit from another.

Table 9: Comparison test with each feature extraction module

Input	Feature Extraction Module	Total features	Accuracy	Rank
Training Sets 15 x 10 numerals	Character Vector Module	35 x 150	64.67%	2
	Kirsch Edge Detection Module	256 x 150	54.00%	3
	Alphabet Profile Feature Extraction Module	10 x 150	28.67%	5
	Modified Character Module	15 x 150	52.00%	4
	Image Compression Module	64 x 150	70.67%	1
	Curvature Vector Module	8 x 150	19.00%	6

From Table 9, it is known that Image Compression Module alone gives the highest recognition accuracy followed by Character Vector Module. However, it is known that by using single feature extraction modules, the accuracy could not reach the target of 90%.

## CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

The handwriting recognition algorithm has long existed and researched, but have yet to achieve a module which is able to produce 100% accuracy due to different handwriting samples and languages. This project not only focused on improving handwritten recognition algorithms by the previous research done by Poo [1], it also implants the concept into an application which is to enhance data entry of examination scores in Universiti Teknologi PETRONAS.

As discussed in the literature review of this progress report, neural network is easy to be used and has proven to give reliable and relevant high accuracy results. Hence, by continuing the previous research, neural network is an avenue to tackle this problem. Sample sets have to be obtained from both students and lecturers as the program has to be robust enough to identify two different trends of handwriting on a single score sheet.

Three tests have been carried out using different feature extraction data to train the neural network. Results have proven as vast amount of improvement of 19.85% in recognition accuracy between the first and second tests. However, with the aim to achieve above 90% accuracy rate, which is 5.48% short, one more feature extraction module, which is Curvature Vector module is added on the third test. The recognition accuracy has finally exceeded the target to 90.67% and a robust data entry system for UTP examination score sheet is being developed

Further progress into this project would include tuning the recognition system to be more robust and accurate can be carried out. Several other feature extraction modules such as data symmetry can be tested to achieve a higher accuracy. A better quality web camera, with higher pixel count should also be used to capture the whole image and overcome the limitations of the current camera. This can help to improve the results of the recognition system as higher pixel count web cameras are able to give much more precise and higher number of bits when the image is being captured and for the system to process.



## REFERENCES

- [1] H. N. Poo, "Handwriting Recognition for Data Entry (HandRec)," University Teknologi PETRONAS, University Teknologi PETRONAS 2006.
- [2] W. L. Goh, D. P. Mital, and H. A. Babri, "An artificial neural network approach to handwriting recognition," in *Knowledge-Based Intelligent Electronic Systems, 1997. KES '97. Proceedings., 1997 First International Conference on*, 1997, pp. 132-136 vol.1.
- [3] K. Pyeoung Kee, "Improving handwritten numeral recognition using fuzzy logic," in *TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, 1997, pp. 539-542 vol.2.
- [4] M. Y. Chen, A. Kundu, and J. Zhou, "Off-line handwritten word recognition (HWR) using a single contextual hidden Markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, 1992, pp. 669-672.
- [5] J. Wan, Y. Huang, G. Zhang, and C. Wan, "Offline Handwritten Numeral Recognition Based on Principal Component Analysis," in *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on*, 2007, pp. 1-298-1-302.
- [6] T. Wakahara, "Shape matching using LAT and its application to handwritten numeral recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, pp. 618-629, 1994.
- [7] C. Zili and W. Zuxue, "A Handwriting Numeral Character Recognition System," in *Multimedia Technology (ICMT), 2010 International Conference on*, 2010, pp. 1-5.
- [8] Y. Fujisawa, S. Meng, T. Wakabayashi, and F. Kimura, "Handwritten numeral recognition using gradient and curvature of gray scale image," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, 1999, pp. 277-280.

- [9] Y. Tai-Shan, T. Yong-Qing, and C. Du-wu, "Research on handwritten numeral recognition method based on improved genetic algorithm and neural network," in *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07. International Conference on*, 2007, pp. 1271-1276.
- [10] W. Xin, T.-l. Huang, and X.-y. Liu, "Handwritten Character Recognition Based on BP Neural Network," in *Genetic and Evolutionary Computing, 2009. WGEN '09. 3rd International Conference on*, 2009, pp. 520-524.
- [11] R. Arnold and P. Miklos, "Character recognition using neural networks," in *Computational Intelligence and Informatics (CINTI), 2010 11th International Symposium on*, 2010, pp. 311-314.
- [12] C. Halder, J. Paul, and K. Roy, "Comparison of the classifiers in Bangla handwritten numeral recognition," in *Radar, Communication and Computing (ICRCC), 2012 International Conference on*, 2012, pp. 272-276.
- [13] P. Zhang, T. D. Bui, and C. Y. Suen, "Hybrid feature extraction and feature selection for improving recognition accuracy of handwritten numerals," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, 2005, pp. 136-140 Vol. 1.
- [14] K. Yamada, H. Kami, J. Tsukumo, and T. Temma, "Handwritten numeral recognition by multilayered neural network with improved learning algorithm," in *Neural Networks, 1989. IJCNN., International Joint Conference on*, 1989, pp. 259-266 vol.2.
- [15] B. V. S. Murthy, "Handwriting recognition using supervised neural networks," in *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, 1999, pp. 2899-2902 vol.4.
- [16] F. Kimura, S. Inoue, T. Wakabayashi, S. Tsuruoka, and Y. Miyake, "Handwritten numeral recognition using autoassociative neural networks," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, 1998, pp. 166-171 vol.1.
- [17] Z. Li and C. Suen, "Distinctiveness and Similarities of Handwritten Numerals," *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE*, vol. 34, pp. 387-396, 2000.

- [18] Y. G. Song, S. Y. Lee, and J. J. Park, "Recognition of Handwritten Numerals Using Multiple Features and Multiple Classifiers," *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE*, vol. 34, pp. 397-405, 2000.
- [19] S. L. Eddins and M. T. Orchard, "Using MATLAB and C in an image processing lab course," in *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, 1994, pp. 515-519 vol.1.

## APPENDICES

### APPENDIX A: TEST 1 MATLAB CODES

#### a. training\_nn.m

```
%Input: 5 sets of sample handwritten numerals to be tested
%Output: A trained neural network system
%Process: Character vector feature extraction

I = imread('sample.bmp');

img = edu_imgpreprocess(I);
for cnt = 1:50
    bw2 = edu_imgcrop(img{cnt});
    charvec = edu_imgresize(bw2);
    out(:,cnt) = charvec;
end

P = out(:,1:50);
T = [eye(10) eye(10) eye(10) eye(10) eye(10)];
Ptest = out(:,1:50);

net = edu_createnn(P,T);
```

#### b. edu\_imgpreprocess.m

```
%Input: Scanned image of handwritten numerals
%Output: Preprocessed binary image
%Process: Converting and plotting the location of objects

function img = edu_imgpreprocess(I)

Igray = rgb2gray(I);
Ibw = im2bw(Igray,graythresh(Igray));
Iedge = edge(uint8(Ibw));
se = strel('square',3);
Iedge2 = imdilate(Iedge, se);
Ifill= imfill(Iedge2,'holes');
[Ilabel num] = bwlabel(Ifill);
Iprops = regionprops(Ilabel);
Ibox = [Iprops.BoundingBox];
Ibox = reshape(Ibox,[4 50]);

Ic = [Iprops.Centroid];
Ic = reshape(Ic,[2 50]);
Ic = Ic';
Ic(:,3) = (mean(Ic.^2,2)).^(1/2);
Ic(:,4) = [1:50];

Ic2 = sortrows(Ic,2);
```

```

for cnt = 1:5
    Ic2((cnt-1)*10+1:cnt*10,:) = sortrows(Ic2((cnt-1)*10+1:cnt*10,:),4);
end

Ic3 = Ic2(:,1:2);
ind = Ic2(:,4);

for cnt = 1:50
    img{cnt} = imcrop(Ibw,Ibox(:,ind(cnt)));
end

```

### c. edu\_imgcrop.m

```

%Input: Preprocessed binary image
%Output: Cropped image to the edge
%Process: Removing the white

function bw2 = edu_imgcrop(bw)

[y2temp x2temp] = size(bw);
x1=1;
y1=1;
x2=x2temp;
y2=y2temp;

cntB=1;
while (sum(bw(:,cntB))==y2temp)
    x1=x1+1;
    cntB=cntB+1;
end

cntB=1;
while (sum(bw(cntB,:))==x2temp)
    y1=y1+1;
    cntB=cntB+1;
end

cntB=x2temp;
while (sum(bw(:,cntB))==y2temp)
    x2=x2-1;
    cntB=cntB-1;
end

cntB=y2temp;
while (sum(bw(cntB,:))==x2temp)
    y2=y2-1;
    cntB=cntB-1;
end

bw2=imcrop(bw,[x1,y1,(x2-x1),(y2-y1)]);

```

#### d. edu\_imgresize.m

```
%Input: Cropped image to the edge
%Output: Image with matrix 5 x 7
%Process: Resizing the binary image

function lett = edu_imgresize(bw2)

bw_7050=imresize(bw2,[70,50]);
for cnt=1:7
    for cnt2=1:5
        Atemp=sum(bw_7050((cnt*10-9:cnt*10),(cnt2*10-9:cnt2*10)));
        lett((cnt-1)*5+cnt2)=sum(Atemp);
    end
end

lett=((100-lett)/100);
lett=lett';
```

#### e. edu\_createnn.m

```
%Input: Image with matrix 5 x 7
%Output: A trained neural network system
%Process: Training the neural network

function net = edu_createnn(P,T)

alphabet = P;
targets = T;

[R,Q] = size(alphabet);
[S2,Q] = size(targets);
S1 = 10;
net = newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;
net.performFcn = 'sse';
net.trainParam.goal = 0.1;
net.trainParam.show = 20;
net.trainParam.epochs = 5000;
net.trainParam.mc = 0.95;
P = alphabet;
T = targets;
[net,tr] = train(net,P,T);
```

#### f. testing\_nn.m

```
%Input: Image with matrix 5 x 7
```

```

%Output: A trained neural network system
%Process: Training the neural network

I = imread('KK.bmp');

img = edu_imgpreprocess(I);
for cnt = 1:50
    bw2 = edu_imgcrop(img{cnt});
    charvec = edu_imgresize(bw2);
    out(:,cnt) = charvec;
end

Ptest = out(:,1:50);
[a,b]=max(sim(net,Ptest));

for cnt = 1:50
    if (b(cnt)==10)
        b(cnt)=0;
    end
end

b = reshape(b,10, []);
b = b'

```

## APPENDIX B: TEST 2 MATLAB CODES

### a. trainnumeral.m

```
%Input: 30 sets of preprocessed image with matrix 1 x 380
%Output: A trained neural network system
%Process: Training the neural network

trainsize = 300;
P = p(:,1:trainsize);

trainsize2=(trainsize/10);
for (eyesize=1:trainsize2)
    T(:, (eyesize-1)*10+1:eyesize*10)=eye(10);
end

net = createnn_2(P,T);
```

### b. createnn\_2.m

```
%Input: Image with matrix 1 x 380
%Output: A trained neural network system
%Process: Training the neural network

function net = createnn_2(P,T)

alphabet = P;
targets = T;

[R,Q] = size(alphabet);
[S2,Q] = size(targets);
S1 = 15;

net = newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');

net.LW{2,1} = net.LW{2,1}*0.001;
net.b{2} = net.b{2}*0.001;
net.performFcn = 'sse';
net.trainParam.goal = 0.0001;
net.trainParam.show = 20;
net.trainParam.epochs = 8000;
net.trainParam.mc = 0.95;
P = alphabet;
T = targets;
[net,tr] = train(net,P,T);
```



### c. testing\_nn\_1.m

```
%Input: Testing samples of numerical handwritings
%Output: Recognized numbers from trained neural network
%Process: Testing the neural network

I = imread('vincent_13.jpg');

Igray = rgb2gray(I);
Ibw = im2bw(Igray,graythresh(Igray));
Iedge = edge(uint8(Ibw));

se = strel('square',2);
Idil=imdilate(Iedge,se);

Ifill = imfill(Idil,'holes');

[Ilabel num] = bwlabel(Ifill);
Iprops = regionprops(Ilabel);
Ibox = [Iprops.BoundingBox];
Ibox = reshape(Ibox,[4 num]);

for cnt = 1:num
    img_crop{cnt} = imcrop(I,Ibox(:,cnt));
end

for cnt = 1:num
    Igray2{cnt} = rgb2gray(img_crop{cnt});
    Ibw2{cnt} = im2bw(Igray2{cnt},graythresh(Igray2{cnt}));
end

noise = 0;
for cnt = 1:num
    if (sum(sum(Ibw2{cnt})) <= 55)
        Ibw2(:,cnt)=[];
        noise = noise + 1;
        num = num - 1;
    end

    if(cnt == num)
        break;
    end
end

for cnt = 1:num
    bw2 = imcrop_1(Ibw2{cnt}); %crop ROI
    bw3=bw2;
    alpha=imresize(bw3,[32 32],'nearest'); %resize image to 32x32
    p1= feautext_1(alpha); %call feature extraction matrix
    out1(:,cnt)= p1; %temporarily save matrixes in out1
    charvec = imgresize_1(bw2); %call character segmentation module
    out(:,cnt) = charvec;
end

p = [out;out1]; %final 380x1 feature matrix
```

```

Ptest = p(:,1:num);
[a,b_ID]=max(sim(net,Ptest));

for cnt = 1:num
    if (b_ID(cnt)==10)
        b_ID(cnt)=0;
    end
end

b_ID

```

#### d. imgcrop\_1.m

```

%Input: Preprocessed binary image
%Output: Cropped image to the edge
%Process: Removing the white

function bw2 = imgcrop_1(bw)

[y2temp x2temp] = size(bw);
x1=1;
y1=1;
x2=x2temp;
y2=y2temp;

cntB=1;
while (sum(bw(:,cntB))==y2temp)
    x1=x1+1;
    cntB=cntB+1;
end

cntB=1;
while (sum(bw(cntB,:))==x2temp)
    y1=y1+1;
    cntB=cntB+1;
end

cntB=x2temp;
while (sum(bw(:,cntB))==y2temp)
    x2=x2-1;
    cntB=cntB-1;
end

cntB=y2temp;
while (sum(bw(cntB,:))==x2temp)
    y2=y2-1;
    cntB=cntB-1;
end

bw2=imcrop(bw,[x1,y1,(x2-x1),(y2-y1)]);

```

### e. `imgresize_1.m`

```
%Input: Cropped image to the edge
%Output: Image with matrix 5 x 7
%Process: Resizing the binary image

function lett = imgresize_1(bw2)

bw_7050=imresize(bw2,[70,50]);
for cnt=1:7
    for cnt2=1:5
        Atemp=sum(bw_7050((cnt*10-9:cnt*10),(cnt2*10-9:cnt2*10)));
        lett((cnt-1)*5+cnt2)=sum(Atemp);
    end
end

lett=((100-lett)/100);
lett=lett';
```

### f. `featext_1.m`

```
%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction of the cropped binary handwritten numbers

function p1 = featext_1(alpha)

h = horin_1(alpha);
v = verti_1(alpha);
d = diagonal_1(alpha);

pro = profiles_1(alpha);

img = imgcomp(alpha);
img = im2col(img,[8 8],'distinct');
[hd,vd,rdd,ldd] = kirsch_1(alpha);
p1 = [h,v,d,hd,vd,rdd,ldd,img',pro]';
```

### g. `horin_1.m`

```
%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using horizontal modified alphabet module

function h = horin_1(alpha)

h1 = [];
h2 = [];
h3 = []; %initialized horizontal matrix

h1 = alpha(1:9,:); %find the h1 region
h1 = sum(h1);
```

```

h1 = h1';
h1 = sum(h1)/288;           %sum all 1

h2 = alpha(10:23,:);      %find the h2 region
h2 = sum(h2);
h2 = h2';
h2 = sum(h2)/448;         %sum all 1

h3 = alpha(24:32,:);      %find the h3 region
h3 = sum(h3);
h3 = h3';
h3 = sum(h3)/288;         %sum all 1

h = [h1 h2 h3];           %horinzontal

```

#### h. verti\_1.m

```

%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using vertical modified alphabet module

```

```
function v = verti_1(alpha)
```

```

v1 = [];
v2 = [];
v3 = [];
v4 = [];
v5 = [];
v6 = [];
v7 = [];
v8 = [];                    %initialized vertical matrix

v1 = alpha(1:13,1:10); %find the v1 region
v1 = sum(v1);
v1 = v1';
v1 = sum(v1)/130;          %sum all 1 divide region size

v2 = alpha(1:19,11:22); %find the v2 region
v2 = sum(v2);
v2 = v2';
v2 = sum(v2)/228;          %sum all 1 divide region size

v3 = alpha(1:13,23:32); %find the v3 region
v3 = sum(v3);
v3 = v3';
v3 = sum(v3)/130;          %sum all 1 divide region size

v4 = alpha(14:32,1:10); %find the v4 region
v4 = sum(v4);
v4 = v4';
v4 = sum(v4)/190;          %sum all 1 divide region size

v5 = alpha(20:32,11:22); %find the v5 region

```

```

v5 = sum(v5);
v5 = v5';
v5 = sum(v5)/156;           %sum all 1 divide region size

v6 = alpha(14:32,23:32); %find the v6 region
v6 = sum(v6);
v6 = v6';
v6 = sum(v6)/190;         %sum all 1 divide region size

v7 = alpha(1:15,20:32); %find the v7 region
v7 = sum(v7);
v7 = v7';
v7 = sum(v7)/195;         %sum all 1 divide region size

v8 = alpha(20:32,1:15); %find the v8 region
v8 = sum(v8);
v8 = v8';
v8 = sum(v8)/195;         %sum all 1 divide region size

v = [v1,v2,v3,v4,v5,v6,v7,v8]; %vertical matrix

```

#### i. diagonal\_1.m

```

%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using diagonal modified alphabet module

function d = diagonal_1(alpha)

d1 = [];
d2 = [];
d3 = [];
d4 = [];           %initialized diagonal matrix

temp = [];
for n = 1:4
    temp = alpha(n,1:n+4);
    d1 = [d1 temp]; %find the d1 region from row 1 to 8
    temp = alpha(n,29-n:32);
    d2 = [d2 temp]; %find the d2 region from row 1 to 8
end

for n = 5:12
    temp = alpha(n,n-4:n+4);
    d1 = [d1 temp]; %find the d1 region from row 9 to 24
    temp = alpha(n,29-n:37-n);
    d2 = [d2 temp]; %find the d2 region from row 9 to 24
end

for n = 13:16
    temp = alpha(n,n-4:16);
    d1 = [d1 temp]; %find the d1 region from row 25 to 32
    temp = alpha(n,17:37-n);
    d2 = [d2 temp]; %find the d2 region from row 25 to 32
end

```

```

end

d1 = d1';
d1 = sum(d1)/size(d1,1);           %sum all 1 divide d1 size
d2 = d2';
d2 = sum(d2)/size(d2,1);           %sum all 1 divide d2 size

for n = 17:20
    temp = alpha(n,29-n:16);
    d3 = [d3 temp];                %find the d3 region from row 33 to 40
    temp = alpha(n,17:n+4);
    d4 = [d4 temp];                %find the d4 region from row 33 to 40
end

for n = 21:28
    temp = alpha(n,29-n:37-n);
    d3 = [d3 temp];                %find the d3 region from row 41 to 56
    temp = alpha(n,n-4:n+4);
    d4 = [d4 temp];                %find the d4 region from row 41 to 56
end

temp = [];
for n = 29:32
    temp = alpha(n,1:37-n);
    d3 = [d3 temp];                %find the d3 region from row 57 to 64
    temp = alpha(n,n-4:32);
    d4 = [d4 temp];                %find the d4 region from row 57 to 64
end

d3 = d3';
d3 = sum(d3)/size(d3,1);           %sum all 1 divide d3 size
d4 = d4';
d4 = sum(d4)/size(d4,1);           %sum all 1 divide d4 size

d = [d1 d2 d3 d4];                 %diago

```

## j. kirsch\_1.m

```

%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using kirsch edge detection module

```

```
function [hd,vd,rdd,ldd] = kirsch_1(alpha)
```

```
alpha = double(alpha);
```

```
mh1 = 1/15 * [ 5  5  5
               -3  0 -3
               -3 -3 -3];
```

```
mh2 = 1/15 * [ -3 -3 -3
                -3  0 -3
                 5  5  5];
```

```

mv1 = 1/15 * [ -3 -3 5
               -3 0 5
               -3 -3 5];

mv2 = 1/15 * [ 5 -3 -3
               5 0 -3
               5 -3 -3];

mrd1 =1/15 * [ -3 5 5
               -3 0 5
               -3 -3 -3];

mrd2 =1/15 * [ -3 -3 -3
               5 0 -3
               5 5 -3];

mld1 =1/15 * [ -3 -3 -3
               -3 0 5
               -3 5 5];

mld2 =1/15 * [ 5 5 -3
               5 0 -3
               -3 -3 -3];

hd1 = conv2(alpha,mh1,'same');
hd2 = conv2(alpha,mh2,'same');
vd1 = conv2(alpha,mv1,'same');
vd2 = conv2(alpha,mv2,'same');
rdd1 = conv2(alpha,mrd1,'same');
rdd2 = conv2(alpha,mrd2,'same');
ldd1 = conv2(alpha,mld1,'same');
ldd2 = conv2(alpha,mld2,'same');

hd = max(hd1,hd2);
vd = max(vd1,vd2);
rdd = max(rdd1,rdd2);
ldd = max(ldd1,ldd2);

%figure(2);
%imshow(hd);
%figure(3);
%imshow(vd);
%figure(4);
%imshow(rdd);
%figure(5);
%imshow(ldd);

hd = imgcomp(hd);
vd = imgcomp(vd);
rdd = imgcomp(rdd);
ldd = imgcomp(ldd);

```

```

hd = im2col(hd,[8 8],'distinct');
vd = im2col(vd,[8 8],'distinct');
rdd = im2col(rdd,[8 8],'distinct');
ldd = im2col(ldd,[8 8],'distinct');

hd = hd';
vd = vd';
rdd = rdd';
ldd = ldd';

```

### k. profiles\_1.m

```

%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using alphabet profile feature extraction

```

```

function pro = profiles_1(alpha)

r = ones(1,32)*32;      %initialize right, top, left , and bottom
feature matrix
t = ones(1,32)*32;
l = ones(1,32)*32;
b = ones(1,32)*32;

for n1 = 1:32
    for n2 = 1:32      %check to find the edge from right boundry
        if alpha(n1,n2) == 1
            r(n1) = n2-1;
            break;      %row 3, 8, 24, 32, 40, 56, 61
        end
    end
end

for n1 = 1:32
    for n2 = 1:32      %check to find the edge from top boundry
        if alpha(n2,n1) == 1
            t(n1) = n2-1;
            break;      %column 3, 8, 24, 32, 40, 56, 61
        end
    end
end

for n1 = 1:32
    for n2 = 1:32      %check to find the edge from left boundry
        if alpha(n1,33-n2) == 1
            l(n1) = n2-1;
            break;      %row 3, 8, 24, 32, 40, 56, 61
        end
    end
end

for n1 = 1:32
    for n2 = 1:32
        if alpha(33-n2,n1) == 1

```



```

        b(n1) = n2-1;
        break;           %column 3, 8, 24, 32, 40, 56, 61
    end
end
end

wid = 32 - [ l(6)+r(6) l(16)+r(16) l(26)+r(26) ];
hei = 32 - [ t(6)+b(6) t(16)+b(16) t(26)+b(26) ];

r = diff(r);
t = diff(t);
l = diff(l);
b = diff(b);

pro = [max(r) max(t) max(l) max(b) wid hei];
pro = pro/32;

```

## 1. imgcomp.m

```

%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x380 matrix
%Process: Feature extraction using image compression module

function out = imgcomp(in)

out = [];

for i = 1:8
    for j = 1:8
        temp = in(4*i-3:4*i,4*j-3:4*j);
        temp = sum(temp);
        out(i,j) = sum(temp')/16;
    end
end
end

```

## APPENDIX C: TEST 3 MATLAB CODES

### a. circle.m

```
%Input: Cropped and resized binary handwritten numbers
%Output: Features in 1x388 matrix
%Process: Feature extraction using curvature vector module

Ori = ~(slantcorrection(scan_image));
Ori = quality(Ori);

%-----
    ---%
a=100; b=100;
[row,column]=size(Ori);
Test=imresize(Ori,[a b]);

%-----add black cover around the image-----
    ---%
Test(:,b+1)=0;
Test(a+1,:)=0;
for i=b:-1:1
    Test(:,i+1)=Test(:,i);
end
Test(:,1)=0;
Test(:,b+2)=0;
for i=a:-1:1
    Test(i+1,:)=Test(i,:);
end
Test(1,:)=0;
Test(a+2,:)=0;
%-----Structure-----
    ---%

c_top=[0,0,0,0]; c_bot=[0,0,0,0];
point=[0,0,0,0,0,0,0,0,0,0,0,0];

%-----choose best column-----
    ---%
oreo = regionprops(Test,'centroid');

best_column = floor(oreo(1,1).Centroid(1));

%-----calculate top circle-----
    ---%
for i=1:a+1
    if c_top(1)~=0 || c_top(2)~=0
        break;
    end
    if Test(i,best_column)==1 && Test((i+1),best_column)==0 && i<44
        point(5)=i+1;
        while Test(point(5),best_column)==1
            point(5)=point(5)+1;
        end
        point(6)=point(5);
    end
end
```

```

while Test(point(5),best_column)==0 && point(5)<102
    for j=best_column:b+1
        if Test(point(5),j)==1 && Test(point(5),(j+1))==1
            c_top(1)=c_top(1)+1;
            break;
        end
    end
    point(5)=point(5)+1;
end
if point(5)==102
    c_top(1)=0;
else
    c_top(2)=point(5)-point(6)-c_top(1);
end
end
end

for i=1:a+1
    if c_top(3)~=0 || c_top(4)~=0
        break;
    end
    if Test(i,best_column)==1 && Test((i+1),best_column)==0 && i<44
        point(7)=i;
        while Test(point(7),best_column)==1
            point(7)=point(7)+1;
        end
        point(8)=point(7);
        while Test(point(7),best_column)==0 && point(7)<102
            for j=best_column:-1:1
                if Test(point(7),j)==1 && Test(point(7),(j-1))==1
                    c_top(3)=c_top(3)+1;
                    break;
                end
            end
            point(7)=point(7)+1;
        end
        if point(7)==102
            c_top(3)=0;
        else
            c_top(4)=point(7)-point(8)-c_top(3);
        end
    end
end
end

%-----calculate bottom circle-----
---%
for i=a+2:-1:2
    if c_bot(1)~=0 || c_bot(2)~=0
        break;
    end
    if Test(i,best_column)==1 && Test((i-1),best_column)==0 && i>70 &&
        i<102
        point(9)=i;
        while Test(point(9),best_column)==1
            point(9)=point(9)-1;
        end
    end
end

```

```

point(10)=point(9);
while Test(point(9),best_column)==0 && point(9)>1
    for j=best_column:b+1
        if Test(point(9),j)==1 && Test(point(9),(j+1))==1
            c_bot(1)=c_bot(1)+1;
            break;
        end
    end
    point(9)=point(9)-1;
end
if point(9)==1
    c_bot(1)=0;
else
    c_bot(2)=point(10)-point(9)-c_bot(1);
end
end
end

for i=a+2:-1:2
    if c_bot(3)~=0 || c_bot(4)~=0
        break;
    end
    if Test(i,best_column)==1 && Test((i-1),best_column)==0 && i>70 &&
        i<102
        point(11)=i;
        while Test(point(11),best_column)==1
            point(11)=point(11)-1;
        end
        point(12)=point(11);
        while Test(point(11),best_column)==0 && point(11)>1
            for j=best_column:-1:1
                if Test(point(11),j)==1 && Test(point(11),(j-1))==1
                    c_bot(3)=c_bot(3)+1;
                    break;
                end
            end
            point(11)=point(11)-1;
        end
        if point(11)==1
            c_bot(3)=0;
        else
            c_bot(4)=point(12)-point(11)-c_bot(3);
        end
    end
end

c_top = c_top / 100;
c_bot = c_bot / 100;
circlefea = [c_top'; c_bot'];

```

## APPENDIX D: TEST 1 TRAINING SET





## APPENDIX F: TEST 3 TRAINING SET

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

# APPENDIX G: TEST SET SAMPLE FOR TEST 1

## TEST SET SAMPLE 1 FOR TEST 1

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
5	6	7	8	9	0	1	2	3	4
1	2	3	4	5	6	7	8	9	0
9	0	1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9	0	1
1	2	3	4	5	6	7	8	9	0
7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3
1	2	3	4	5	6	7	8	9	0

Results:

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
5	6	7	8	9	0	1	2	3	4
1	2	3	4	5	6	7	8	9	0
9	0	1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9	0	1
1	2	3	4	5	6	7	8	9	0
7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3
1	2	3	4	5	6	7	8	9	0



## TEST SET SAMPLE 2 FOR TEST 1



Results:

6	2	7	4	5	6	7	1	9	0
1	7	1	7	5	6	7	6	9	0
1	1	1	4	5	6	7	6	9	0
1	7	7	4	5	8	7	6	9	0
1	2	1	1	5	6	7	8	9	0
1	8	3	4	5	6	7	8	9	0
1	2	5	1	5	6	7	8	9	0
8	6	5	9	5	6	7	8	9	0
1	8	3	3	5	6	7	6	9	0
8	2	8	7	5	6	7	1	9	0
4	2	5	4	8	0	7	5	5	0
1	3	5	7	5	6	7	6	7	0
8	2	5	4	5	6	5	1	0	0
1	1	7	4	5	0	7	8	7	0
3	2	5	4	5	6	7	6	9	0

## APPENDIX H: TEST SET SAMPLE FOR TEST 2

### TEST SET SAMPLE 3 FOR TEST 2

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4.5	6	7	8	9	0	
1	2	3	4.5	6	7	8	9	0	
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

Results:

1	2	3	4	3	6	7	8	9	0
1	3	3	4	3	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	3	3	4	5	6	7	8	9	0
1	2	3	4	3	6	7	8	7	0
1	2	3	4	5	6	7	8	9	0
1	2	7	4	5	6	7	2	9	6
7	2	2	2	3	6	7	2	9	7
1	2	3	4	2	6	7	5	7	0
1	1	2	4	3	0	7	8	9	0
1	2	3	9	5	6	7	8	9	0
1	2	3	4	5	6	7	2	9	0
1	2	3	4	3	6	7	9	9	0
1	2	3	4	5	6	7	5	9	0
5	2	3	4	5	6	7	8	9	0

## TEST SET SAMPLE 4 FOR TEST 2

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
3	4	1	2	7	8	9	3	1	2
0	9	8	7	6	5	4	3	2	1
8	8	8	8	1	2	0	0	3	2
1	0	1	0	1	0	1	0	1	1
2	1	3	0	1	1	0	1	1	4
6	8	7	3	4	2	5	9	0	1
1	9	0	9	1	9	9	2	3	3
5	2	0	1	3	1	4	0	2	5
9	3	9	3	1	7	7	7	3	7
6	9	6	9	6	9	6	9	6	9
1	3	5	7	9	2	4	6	8	0
2	4	6	8	0	1	3	5	7	9

Results:

1	2	3	4	3	6	7	8	9	0
2	2	3	4	4	0	7	8	9	0
1	2	3	4	3	6	7	8	9	0
3	4	1	2	7	8	9	3	2	2
0	9	8	7	6	3	4	3	2	1
8	8	8	8	1	2	0	0	3	2
1	0	1	0	1	0	1	0	1	7
2	4	3	0	1	1	0	1	1	4
6	8	7	3	4	2	5	9	0	1
9	9	0	9	1	9	9	2	3	3
3	2	0	1	3	7	4	0	2	5
9	3	9	3	1	9	7	7	3	7
6	9	6	9	6	9	6	9	6	9
7	3	3	7	9	2	4	6	8	0
2	4	6	8	0	1	3	3	7	9

# APPENDIX I: TEST SET SAMPLE FOR TEST 3

## TEST SET SAMPLE 5 FOR TEST 3

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

Results:

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	3	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	9	8	9	0
1	1	3	4	5	6	4	8	9	0
1	2	4	4	5	6	9	8	9	0
1	2	3	4	5	6	9	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	3	3	4	5	6	7	8	9	0
1	2	3	4	3	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	1	1	4	5	6	9	8	9	0
1	2	3	4	5	6	7	8	9	0
1	6	3	4	5	6	4	8	9	1
1	2	3	4	3	6	9	8	8	0
1	2	3	4	5	6	9	8	9	0
1	2	3	4	3	6	8	8	9	0
1	2	3	4	3	1	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	3	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	4	3	4	3	6	7	1	9	0

### TEST SET SAMPLE 6 FOR TEST 3

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0

Results:

1	2	3	7	5	4	7	8	9	0
1	2	3	4	5	6	9	8	9	0
7	2	3	4	5	6	4	8	9	0
1	2	3	4	3	6	9	1	9	0
1	2	9	4	3	6	7	8	7	0
1	2	3	4	9	6	4	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	9
1	2	3	4	5	6	7	3	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	9	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	3	6	9	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0
1	4	3	4	7	6	7	8	9	0
1	2	3	4	5	6	9	8	9	0
1	2	3	4	9	6	4	8	7	0
1	2	3	4	5	9	7	8	9	0
1	4	3	4	5	6	7	8	9	0
1	2	4	4	5	8	8	8	9	0

## APPENDIX J: SEDEX AWARDS







UNIVERSITI  
TEKNOLOGI  
PETRONAS



SCIENCE & ENGINEERING  
DESIGN EXHIBITION

# *Certificate of* ACHIEVEMENT

THIS IS TO CERTIFY THAT  
**WONG YOONG XIANG**

HAS BEEN AWARDED THE

**IDEA GENERATED FUND AWARD  
"EDUCATION"**

IN THE

**34<sup>TH</sup> EDITION OF  
SCIENCE & ENGINEERING DESIGN EXHIBITION  
(SEDEX 34)**

HELD ON

**8 & 9 DECEMBER 2014**

AT

**UNIVERSITI TEKNOLOGI PETRONAS**

PROF. IR. DR. AHMAD FADZIL MOHAMAD HANI  
DEPUTY VICE CHANCELLOR (ACADEMIC)  
UNIVERSITI TEKNOLOGI PETRONAS