UNIVERSITY OF TECHNOLOGY

(YATANARPON CYBER CITY)

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE

HANDWRITING RECOGNITION FOR MYANMAR CONSONANTS USING NEURAL NETWORK

BY

SHIN THANT

B.E. THESIS

SEPTEMBER, 2018

UNIVERSITY OF TECHNOLOGY
(YATANARPON CYBER CITY)
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION SCIENCE

HANDWRITING RECOGNITION FOR MYANMAR CONSONANTS USING NEURAL NETWORK

BY
SHIN THANT

A partial dissertation submitted to the University of Technology, Yatanarpon Cyber City in fulfillment of the requirement for the degree of

Bachelor of Engineering

(Information Science & Technology)

19th SEPTEMBER, 2018

DECLARATION

I hereby declare that I carried out the work reported in this thesis in the Department of Information Science, Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), under the supervision of Dr. Tin Myint Naing. I solemnly declare that to the best of my knowledge, no part of this thesis has been submitted here or elsewhere in a previous application for award of a degree. All sources of knowledge used have been duly acknowledged.

……

19th September 2018

SHIN THANT

6IST-17

APPROVAL

This is to certify that the B.E. thesis titled "**HANDWRITING RECOGNITION FOR MYANMAR CONSONANTS USING NEURAL NETWORK**" carried out by **SHIN THANT**, **6IST-17**, has been read and approved for meeting part of the requirements and regulations governing the award of the degree of Bachelor of Engineering (Information Science and Technology), Department of Information Science under the Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), Myanmar.

1. Dr. Hnin Aye Thant
   Professor and Head
   Department of Information Science ……
   Faculty of Information and Communication Technology (Chairman)

2. Dr. Phyu Phyu Tar
   Professor
   Department of Information Science ……
   Faculty of Information and Communication Technology (Course Coordinator)

3. Dr. Tin Myint Naing
   Associate Professor
   Department of Information Science ……
   Faculty of Information and Communication Technology (Supervisor)

4. Dr. Htet Ne Oo
   Lecturer
   Department of Information Science ……
   Faculty of Information and Communication Technology (Member)

5. Dr. Nandar Win Min
   Assistant Lecturer
   Department of Information Science ……
   Faculty of Information and Communication Technology (Member)

ACKNOWLEDGMENTS

First of all, I would like to express my special thanks to Dr. Aung Win, Rector, University of Technology (Yatanarpon Cyber City), for initiating the Bachelor programme at the University of Technology (Yatanarpon Cyber City).

I would like to express grateful thank to Dr. Hnin Aye Thant, Professor, Head of Department of Information Science, Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), for her kind guidance, encouragement.

I am very grateful to Dr. Phyu Phyu Tar, Professor and Course Coordinator of Information Science and Technology, Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), for her arrangement, valuable suggestions, comments and advice in completion of this thesis.

I am very grateful to my supervisor, Dr. Tin Myint Naing, Associate Professor of Department of Information Science, Faculty of Information and Communication Technology, University of Technology (Yatanarpon Cyber City), for his kind guidance and encouragement in making my thesis to complete successfully. He has been very supportive in this B.E. Thesis, and also guides a lot, particularly, at the level of quality of presentation.

Very special thanks to Daw Moe Pale, Lecturer of English Department, University of Technology (Yatanarpon Cyber City) for her valuable supports from the language point of view in my B.E. Thesis work.

I would like to thank a lot to all my teachers for their mentoring, encouragement, and recommending this dissertation.

Finally, I am grateful to my parents and friends who specially offered strong moral and physical support, care and kindness, during the year of my B.E. Thesis study.

ABSTRACT

In education field, android-based applications can be used as teaching aids that can provide a lot of benefits. To practice writing Myanmar consonants, kids will need paper, pen and a teacher to check in traditional way. This system is implemented as an android application providing children to learn thirty three Myanmar consonants. Kids can practice consonants by writing on phone screen. The words can be automatically checked whether it is correct or not. Then the result is shown to the users. This system will help kids to enjoy studying and it is also useful for illiterates. To capture writing word, signature pad library is used in android studio. In this system, back-propagation neural network approach is applied to recognize the Myanmar consonants. The training data is obtained by performing binary conversion step and segmentation step using projection profile method. Features are extracted using histogram of oriented gradients (HOG) method. This system is implemented using android and matlab programming.

CONTENTS

	Page
DECLARATION	i
APPROVAL	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF EQUATIONS	x
CHAPTER 1 INTRODUCTION	
1.1 Objectives	1
1.2 Field Background	2
1.2.1 Image Preprocessing	2
1.2.2 Image Segmentation	2
1.2.3 Feature Extraction	3
1.2.4 Classification	3
1.2.5 Android Software Development	4
1.3 Overview of the System	4
1.4 Organization of the Thesis	5
CHAPTER 2 THEORETICAL BACKGROUND	
2.1 Image Preprocessing	6
2.1.1 RGB to Grey-scale Image Conversion	6
2.1.2 Thresholding	7
2.1.3 Mean Filtering	7
2.2 Image Segmentation	7
2.2.1 Projection Histogram	8
2.2.1.1 Projection Process	9
2.3 Feature Extraction	9
2.3.1 Histogram of Oriented Gradients	9
2.3.1.1 HOG Algorithm Overview	10
2.3.1.2 Gradient Computation	11

2.3.1.3 Orientation Binning 12
2.3.1.4 Descriptor Blocks 13
2.3.1.5 Block Normalization 14
2.4 Neural Network 15
2.4.1 Components of Neural Network 16
2.4.1.1 Neurons 16
2.4.1.2 Connections and Weights 17
2.4.1.3 Activation Function 17
2.4.1.4 Learning Rule 18
2.4.2 Architecture of Neural Network 18
2.4.3 Learning Paradigms 19
2.4.3.1 Supervised Learning 19
2.4.3.2 Unsupervised Learning 20
2.4.3.3 Reinforcement Learning 20
2.4.4 Learning Process 21
2.4.5 Multilayer Neural Network 22
2.4.6 Back-propagation Neural Network 23
2.4.6.1 Back-propagation Training Algorithm 24
2.4.6.2 Application Categories of the Back-propagation Model 26
2.4.7 Advantages and Disadvantages of Neural Network 27
2.5 Android Software Development 28
2.5.1 Android SDK 28
2.5.2 Android Library 29
2.5.3 Android Signature Pad 29
CHAPTER 3 SYSTEM DESIGN AND IMPLEMENTATION
3.1 System Design 30
3.2 System Flow 31
3.3 System Implementation 31
3.4 Accuracy of the System 41
3.5 Output of the System 43
CHAPTER 4 CONCLUSION
4.1 Advantages of the System 44
4.2 Limitation of the System 44

4.3 Further Extension 45
REFERENCES 46

LIST OF FIGURES

Figure		Page
2.1	Text Segmentation using Projection Profile	8
2.2	Sample HOG Visualization of Number Three	10
2.3	HOG Algorithm Flow	11
2.4	Computation of Gradient Magnitude and Direction of a Cell	12
2.5	Computed Gradient and Orientation-based Histogram Channel	13
2.6	R-HOG Block and C-HOG Block	14
2.7	Components of Neural Network	16
2.8	Some Activation Functions of Neuron	17
2.9	Process of Supervised Learning	19
2.10	Process of Unsupervised Learning	20
2.11	Process of Reinforcement Learning	21
2.12	Architecture of Multilayer Neural Network	23
2.13	Architecture of Back-propagation Neural Network	24
3.1	System Design	30
3.2	System Flow	32
3.3	Welcome Screen	33
3.4	Home Page	33
3.5	Main Study Page	34
3.6	Study Page	35
3.7	Practice Page	35
3.8	Data Stored in Firebase Storage	36
3.9	Classification of the Acquired Image	36
3.10	Web Page in Php	37
3.11	Practice Page with Correct Result	38
3.12	Practice Page with Wrong Result	38
3.13	Exit Page	39
3.14	Home Page in Matlab	39
3.15	Second Page in Matlab	40
3.16	Testing Page in Matlab	40
3.17	Testing Page with Tested Data in Matlab	41

LIST OF TABLES

Table		Page
2.1	Analogy between Biological and Artificial Neural Networks	15
3.1	Confusion Matrix for the Accuracy of the System	41

LIST OF EQUATIONS

Equation		Page
2.1	Projection Histogram	8
2.2	Magnitude	11
2.3	Direction	11
2.4	L2-norm	14
2.5	L1-norm	14
2.6	L1-sqrt	14
2.7	Activation Function	16
2.8	Output Function	16
2.9	Input Function	17
2.10	Random Weight and Threshold Levels	24
2.11	Hidden Layer Actual Output	25
2.12	Output Layer Actual Output	25
2.13	Error Gradient in Output Layer	25
2.14	Error in Output Layer	25
2.15	Weight Correction in Output Layer	25
2.16	New Weight in Output Layer	25
2.17	Error Gradient in Hidden Layer	25
2.18	Weight Correction in Hidden Layer	25
2.19	New Weight in Hidden Layer	26
3.1	Accuracy	42

CHAPTER 1
INTRODUCTION

In the present age, Information Technology (IT) is becoming a vital part in human lives. As technology can provide ways to make traditional works easier, it is involved in just about everything human does in modern society. So IT can be applied in many fields as education, science, business, health, entertainment and social for a better purpose. In education field, IT can provide more effective ways of learning and teaching. The effective use of technology in Education has changed the face of education and it has created more educational opportunities. Both teachers and students have benefited from various educational technologies.

Information technology includes various fields such as Operations Research (OR), Artificial Intelligence (AI), Software Engineering (SE), Networking, etc. The field of Artificial Intelligence is concerned with methods of developing systems that display aspects of intelligence behavior. Artificial intelligence is the branch of computer science concerned with making computers behave like humans. A neural network model which is the branch of artificial intelligence is generally referred to as artificial neural networks (ANNs). Neural network is widely used in recognition and classification tasks. ANN is to find the most appropriate grouping of training, learning and transfer function for classifying the data sets with growing number of features and classified sets.

Mobile applications can also be implemented to be a part of educational technology. An application that is intended for children and illiterates is designed to practice and learn thirty-three Myanmar consonants. The application allows users to do two parts. The first part is that the users can study each of the thirty-three consonants together with their pronunciation and ways of how to write. The second part is that the users can practice writing. The system will allow users to learn Myanmar consonants without teachers, paper, and pens as traditional way. ANN is used to recognize the consonants in the system.

1.1 Objectives

The objectives of the thesis are as follows:

- To provide the effective usage of technology for kids
- To support an attractive learning environment for kids
- To provide flexible learning environment for illiterates
- To make more efficient way of learning Myanmar consonants

1.2 Field Background

In handwriting recognition system, the core task, recognition process is performed using neural network field. According to the procedure of neural network, input features are required to train the network. The required features are obtained using histogram of oriented gradients (HOG) method. To obtain the input to HOG method, the rough image is acquired after performing image preprocessing and segmentation steps. This is the overview of the applied fields in the system.

1.2.1 Image Preprocessing

As a general first step in a recognition system, preprocessing plays a very important role and can directly affect the recognition performance [2]. The purpose of preprocessing is to discard irrelevant information in the input data, which can negatively affect the recognition [9]. Preprocessing usually consists of gray conversion, filtering and binarization.

In this system, the training or testing image is firstly converted to gray image. Then the original training or testing image is converted to binary image with threshold value of the gray image. Then the training image is segmented and testing image is cropped. Then the segmented or cropped image is filtered using mean filtering in order to remove noise.

1.2.2 Image Segmentation

Segmentation occupies a very important role in image processing because it is so often the vital first step which must be successfully taken before subsequent tasks such as feature extraction, classification, description, etc. can be sensibly attempted. The basic goal of segmentation is to partition the image into mutually exclusive regions to which segments can be subsequently attached meaningful labels. Image segmentation is usually approached through one of the two basic routes. The two basic routes are edge or boundary-based methods and region-based methods.

In this system, region-based segmentation method, projection histogram is used as this method is mostly used for segmentation of lines, words and characters. In the process, the document is segmented into lines first, followed by word and character segmentation. This method involved two parts: horizontal projection (line segmentation) and vertical projection (word segmentation). Horizontal projection is the total number of all black-pixels along a column while vertical projection is the total number of all black-pixels along a row of a character image.

1.2.3 Features Extraction

An image feature is a distinguishing primitive characteristic or attribute of an image. Features can be extracted based on shape, color, and texture. The purpose of feature extraction is to extract more useful/dominant information hidden in the signals by avoiding unnecessary or redundant information. Image is pre-processed to remove noise, interferences and artifacts before performing feature extraction.

In this system, Histogram of Oriented Gradients (HOG) is used as this method is scale invariant and rotation variant. The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. It represents objects as a single feature vector as opposed to a set of feature vectors where each represents a segment of the image. It is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. It is computed by sliding window detector over an image, where a HOG descriptor is computed for each position. Therefore, it is easy to express the rough shape of the object and is robust to variations in geometry and illumination changes. On the other hand, rotation and scale changes are not supported.

1.2.4 Classification

Classification is the problem of identifying a new observation belongs to which of a set of categories, on the basis of a training set of data containing observations (or instances) whose category membership is known.

In this system, back-propagation neural network is used as it is the most widely used for classification. As with any other neural network, a back-propagation one is determined by the connections between neurons (the network's architecture),

the activation function used by the neurons, and the learning algorithm (or the learning law) that specifies the procedure for adjusting weights.

In a back-propagation neural network, the learning algorithm has two phases. First, a training input pattern is presented to the network input layer. The network then propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated. The back-propagation model is called supervised learning method.

1.2.5 Android Software Development

Android software development is the process by which new applications are created for devices running the Android operating system. Android apps can be written using Kotlin, Java, and C++ languages using the Android software development kit (SDK), while using other languages is also possible.

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes required libraries, debugger, an emulator, relevant documentation for the Android application program interfaces (APIs), sample source code and tutorials for the Android OS.

An Android library is structurally the same as an Android app module. It can include everything needed to build an app, including source code, resource files, and an Android manifest. Android Signature Pad is an Android library for drawing smooth signatures. This library is mainly used to capture the writing consonants from users.

1.3 Overview of the System

This thesis title is “Handwriting Recognition for Myanmar Consonants using Neural Network”. This system is implemented to learn writing and pronunciation of thirty-three Myanmar consonants.

This system involves two parts. As the first part, the users can study each consonant with the respective writing and pronunciation. They can study consecutively or randomly based on the consonant they desired to study. In the second part, the users can practice the writing of consonant according to the pronunciation of

random consonant. Then the written consonant is checked whether it is the same or not as the random one. If the written consonant is correct, the message ‘Correct Answer’ will appear together with clapping voice and happy emoji. If the written consonant is wrong, the message ‘Wrong Answer’ will appear together with not only failure voice and sad emoji, the system will show the correct answer.

This system is an application using Android Programing and Matlab Programing. Firebase storage and xampp server are also used to establish connection between android and matlab. Application name is “MMConsonant”. One hundred and twenty data for each consonant is used as training data. To create this application, Android Studio IDE and Matlab IDE are used.

1.4 Organization of the Thesis

In this thesis, there are four chapters.

Chapter one describes introduction, objectives, overview of the system, and organization of the thesis.

Chapter two discusses about theoretical background in detail. It represents about segmentation method (Projection Profile), feature extraction method (Histogram of Gradients), classification method (Neural Network) and android software development.

Chapter three is “Design and Implementation”. This chapter represents the system design and implementation, system flow, output of the system using neural network.

Chapter four describes the conclusion, expected output, benefits, limitation and further extension of the thesis.

CHAPTER 2
THEORETICAL BACKGROUND

This chapter presents the detail of the theory background that is applied in the system. Back-propagation neural network is applied to recognize handwriting consonants. The required inputs to train neural network are obtained after preprocessing, segmentation and feature extraction steps. The input image is segmented by using projection profile method and features are extracted by using histogram of oriented gradients (HOG) method. To develop android application, android studio IDE is used. As the system need to acquire the written word, Android Signature Pad library is used.

2.1 Image Preprocessing

As a general first step in a recognition system, preprocessing can directly affect the recognition performance [1]. The purpose of preprocessing is to discard irrelevant information in the input data, which can negatively affect the recognition [9]. Preprocessing usually consists of gray conversion, filtering and binarization.

In this system, the training or testing image is firstly converted to gray image. Then the original training or testing image is converted to binary image with threshold value of the gray image. Then the training image is segmented and testing image is cropped. Then the segmented or cropped image is filtered using mean filtering in order to remove noise.

2.1.1 RGB to Grey-scale Image Conversion

Grey-scale conversion is the initial step in many image analysis algorithm, as it essentially simplifies (i.e. reduces) the amount of information in the image. Although a grey-scale image contains less information than a color image, the majority of important, feature-related information is maintained, such as edges, regions, blobs, junctions and so on. Feature detection and processing algorithms then typically operate on the converted grey-scale version of the image. RGB to grey-scale conversion is a noninvertible image transform: the true color information that is lost in the conversion cannot be readily recovered [2].

2.1.2 Thresholding

Thresholding produces a binary image from a grey-scale or color image by setting pixel values to 1 or 0 depending on whether they are above or below the threshold value. This is commonly used to separate or segment a region or object within the image based upon its pixel values.

In its basic operation, thresholding operates on an image *I* as follows: [2]

```
for each pixel I (i, j) within the image I
    if I (i, j) > threshold
        I (i, j) = 1
    else
        I (i, j) = 0
    end
end
```

2.1.3 Mean Filtering

The mean filtering is perhaps the simplest linear filter and operates by giving equal weight w_k to all pixels in the neighborhood. A weight of $W_K = 1/(NM)$ is used for an N × M neighborhood and has the effect of smoothing the image, replacing every pixel in the output image with the mean value from its N × M neighborhood. This weighting scheme guarantees that the weights in the kernel sum to one over any given neighborhood size. Mean filters can be used as a method to suppress noise in an image. Another common use is as a preliminary processing step to smooth the image in order that some subsequent processing operation will be more effective.

The main drawbacks of mean filtering are (a) it is not robust to large noise deviations in the image (outliers) and (b) when the mean filter straddles an edge in the image, it will cause blurring. For this latter reason, the mean filter can also be used as a general low-pass filter [2].

2.2 Image Segmentation

Segmentation is the generic process by which an image is subdivided into its constituent regions or objects. Segmentation occupies a very important role in image processing because it is so often the vital first step which must be successfully taken before subsequent tasks such as feature extraction, classification, description, etc. can

be sensibly attempted. The basic goal of segmentation is to partition the image into mutually exclusive regions to which segments can be subsequently attached meaningful labels. The segmented objects are often termed as the foreground and the rest of the image was the background. The correct segmentation of the image is depended strongly on the types of objects which are interested in identifying. In this system, region-based segmentation method, projection histogram is used.

Image segmentation is usually approached through one of two basic routes:

- Edge/boundary methods: This approach is based on the detection of edges as a means to identifying the boundary between regions. As such, it looks for shape differences between groups of pixels.
- Region-based methods: This approach assigned pixels to a given region based on their degree of mutual similarity [2].

2.2.1 Projection Histogram

Projection histograms were introduced in 1956 in a hardware OCR system by Glauberman. It consists of a simple running count of the black-pixels in one direction, i.e. a column or a row. Today, this technique is mostly used for segmentation of lines, words and characters. In the process, the document is segmented into lines first, followed by word and character segmentation.

Basically two types of projections are used: horizontal and vertical. Horizontal projection was the total number of all black-pixels along a column while vertical projection was the total number of all black-pixels along a row of a character image. An example of segmenting an image with hundred digits using projection method is shown in figure 2.1. For example, if there is n number of rows, then vertical projection:

$$Y = \sum A_{r_i} \tag{2.1}$$

, where A is for black-pixel, r_i is the row number; i = 1, 2, 3… n. [4]

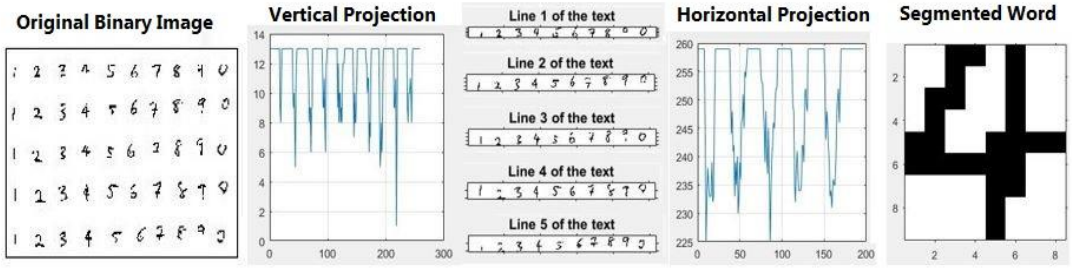

Figure 2.1 Text Segmentation using Projection Profile

2.2.1.1 Projection Process

The image is binarized first. To segment each line from image, pixels in the image are summed horizontally. Then distinguish the lines that have text and the lines that do not have text based on threshold. Next step was to identify the top pixel and bottom pixel of each line. This step would provide each line separately from the image. This is called 'vertical projection'.

For each line obtained from vertical projection, the pixels are summed vertically. Then distinguish again the words that have text and the space between words based on threshold. Next step was to identify the leftmost pixel and rightmost pixel of each word. This step will provide each word separately from the image. This is called 'horizontal projection'. In this way, each word in the image can be segmented successfully by using 'Projection Profile'.

2.3 Feature Extraction

An image feature is a distinguishing primitive characteristic or attribute of an image. Some features are natural in the sense that such features are defined by the visual appearance of an image, while other, artificial features result from specific manipulations of an image. Features can be extracted based on shape, color, and texture. In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived features intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

The purpose of feature extraction is not only to reduce the dimensionality but also to extract more useful/dominant information hidden in the signals by avoiding unnecessary or redundant info. Image is pre-processed to remove noise, interferences and artifacts before performing feature extraction. After features are extracted, Classification is performed based on the selected features. The performance of classifier depends on both how good the signals are pre-processed and how good the features are extracted.

2.3.1 Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.

It represents objects as a single feature vector as opposed to a set of feature vectors where each represents a segment of the image. It is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. It is computed by sliding window detector over an image, where a HOG descriptor is a computed for each position. Therefore, it is easy to express the rough shape of the object and is robust to variations in geometry and illumination changes. On the other hand, rotation and scale changes are not supported.

The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ('cells'), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram 'energy' over somewhat larger spatial regions ('blocks') and using the results to normalize all of the cells in the block. The normalized descriptor blocks were be referred as Histogram of Oriented Gradient (HOG) descriptors [5]. Figure 2.2 shows a simple example of histogram of oriented gradients visualization in digit recognition.

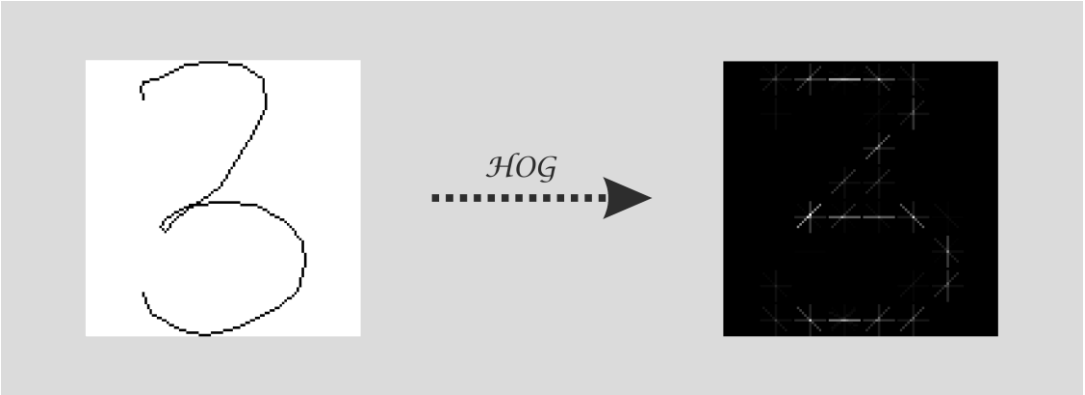

Figure 2.2 Sample HOG Visualization of Number Three

2.3.1.1 HOG Algorithm Overview

1) Divide image into small sub-images: "cells" (Cells can be rectangular(R-HOG) or circular(C-HOG)
2) Accumulate a histogram of edge orientations within that cell

3) The combined histogram entries are used as the feature vector describing the object
4) To provide better illumination invariance (lighting, shadows, etc.) normalize the cells across larger regions incorporating multiple cells: "blocks" [6].

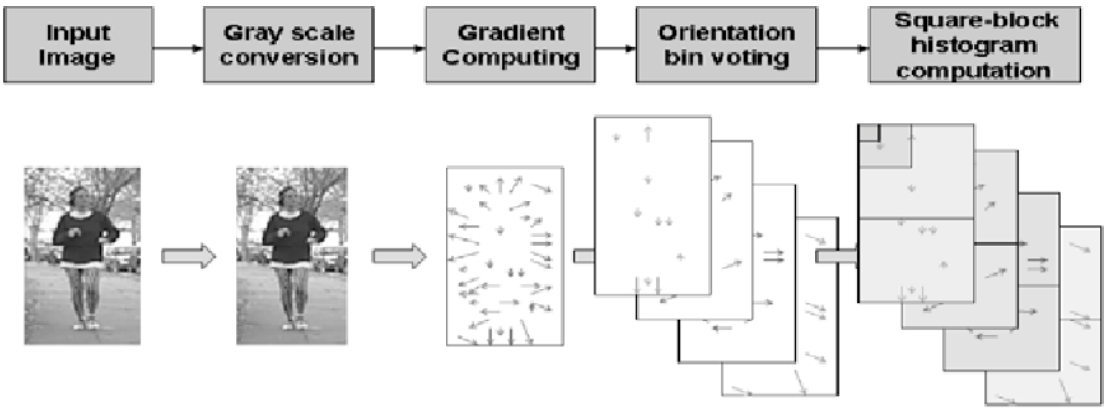

Figure 2.3 HOG Algorithm Flow [14]

2.3.1.2 Gradient Computation

The first step of calculation in many feature detectors in image pre-processing is to ensure normalized color and gamma values. This step can be omitted in HOG descriptor computation, as the ensuing descriptor normalization essentially achieves the same result. Image pre-processing thus provides little impact on performance.

Instead, the first step of calculation is the computation of the gradient values. The most common method is to apply the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. Specifically, this method requires filtering the color or intensity data of the image with following filter kernels:

$$[-1, 0, 1] \text{ and } [-1, 0, 1]^T$$

In fact, there are many more complex masks, such as Sobel, Prewitt, Canny or diagonal masks, but these masks generally result in poorer performance. Simple 1-D [-1; 0; 1] masks at σ=0 work best. Then, the magnitude and orientation at each pixel I(x, y) is calculated by

$$G_{mag}(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{2.2}$$

$$\theta(x,y) = arctan\left(G_y(x,y)/G_x(x,y)\right) + \pi/2 \tag{2.3}$$

where $G_x(x,y)$ and $G_y(x,y)$ are the gradient values at each pixel in horizontal and vertical direction, respectively.

For color images, separate gradients for each color channel is calculated, and the one with the largest norm is taken as the pixel's gradient vector. It should be noted that π=2 is necessary since the arctan operator results in a range between -π=2 and π=2, but for unsigned orientation scheme which gives better performance, it ranges from 0 to π [8].

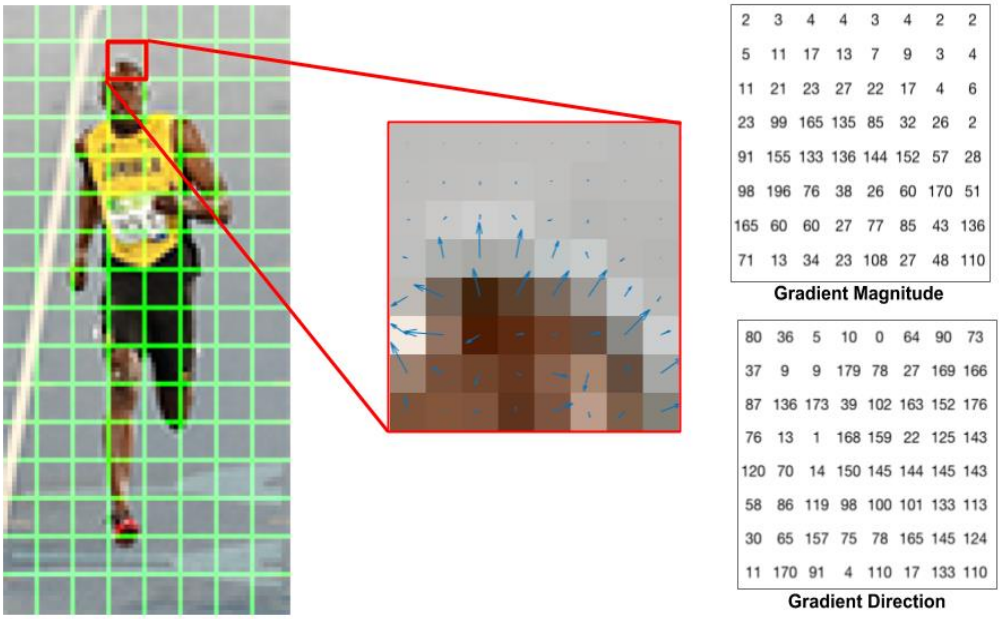

Figure 2.4 Computation of Gradient Magnitude and Direction of a Cell [13]

2.3.1.3 Orientation Binning

The second step of calculation is creating the cell histograms. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is “unsigned” or “signed”. So, every histogram bin has a spread of 20 degrees with nine or eighteen histogram channels.

The unsigned gradients used in conjunction with the nine histogram channels performed best in the human detection experiments. As for the vote weight, pixel contribution can either be the gradient magnitude itself, or some function of the magnitude. Generally, the gradient magnitude itself generally produces the best

results. Other options for the vote weight could include the square root or square of the gradient magnitude, or some clipped version of the magnitude [8].

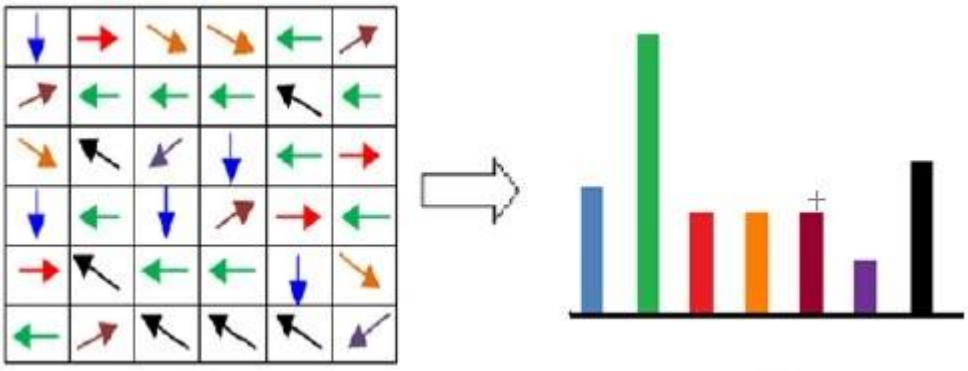

Figure 2.5 Computed Gradient and Orientation-based Histogram Channel

2.3.1.4 Descriptor Blocks

To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the concatenated vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular R-HOG blocks and circular C-HOG blocks.

R-HOG blocks are generally square grids, represented by three parameters:

1) The number of cells per block,
2) The number of pixels per cell, and
3) The number of channels per cell histogram.

In the Dalal and Triggs human detection experiment, the optimal parameters were found to be four 8x8 pixels cells per block (16x16 pixels per block) with 9 histogram channels.

Circular HOG blocks (C-HOG) can be found in two variants: those with a single, central cell and those with an angularly divided central cell. In addition, these C-HOG blocks can be described with four parameters:

1) The number of angular bins,
2) The number of radial bins,
3) The radius of the center bin, and
4) The expansion factor for the radius of additional radial bins.

The two main variants provided equal performance, and that two radial bins with four angular bins, a center radius of 4 pixels, and an expansion factor of 2 provided the best performance in the experimentation (to achieve a good performance, at last use this configure). Also, Gaussian weighting provided no benefit when used in conjunction with the C-HOG blocks. C-HOG blocks appear similar to shape context descriptors, but differ strongly in that C-HOG blocks contain cells with several orientation channels, while shape contexts only make use of a single edge count in their formulation [8]. Figure 2.6 shows simple R-HOG block and C-HOG block.

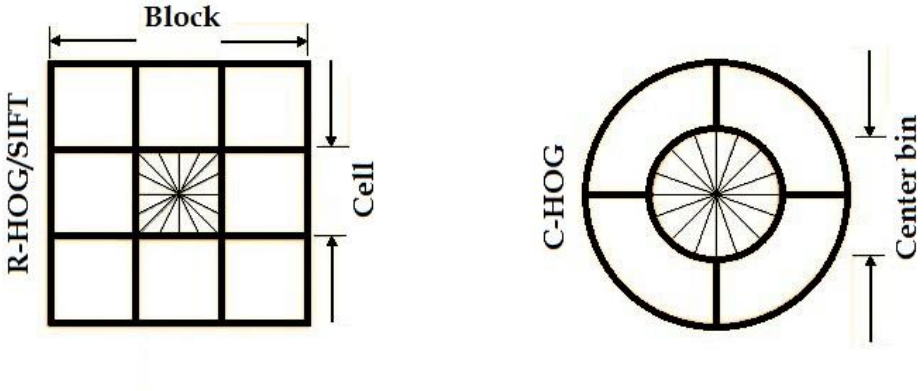

Figure 2.6 R-HOG Block and C-HOG Block

2.3.1.5 Block Normalization

There are four different methods proposed for block normalization. Let v be the non-normalized vector containing all histograms in a given block, $\| v \|_k$ be its k-norm for k = 1, 2 and 'e' be some small constant (the exact value, hopefully, is unimportant). Then the normalization factor can be one of the following:

L2-norm: $f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$ (2.4)

L1-norm: $f = \frac{v}{(\|v\|_1 + e)}$ (2.5)

L1-sqrt: $f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$ (2.6)

In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. The L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less

reliable performance; however, all four methods showed very significant improvement over the non-normalized data. The final HOG feature descriptor is then the vector containing the elements of the normalized cell histograms from all of the block regions [8].

2.4 Neural Network

A neural network (NN) is an abstract computer model of the human brain. The human brain has an estimated 1011 tiny units called neurons. These neurons are interconnected with an estimated 1015 links. Although more research need to be done, the neural network of the brain is considered to be the fundamental source of intelligence, which includes perception, cognition, and learning for humans as well as other living creatures.

Although the term “neural networks” (NNs) is most commonly used, other names include artificial neural networks (ANNS) to distinguish from the natural brain neural networks-neural nets, PDP (Parallel Distributed Processing) models (since computations can typically be performed in both parallel and distributed processing) connectionist models, and adaptive systems [7].

An artificial neural network consists of a number of very simple and highly interconnected processors, also called neurons, which are analogous to the biological neurons in the brain. The output signal is transmitted through the neuron's outgoing connection (corresponding to the biological axon). The outgoing connection, in turn, splits into a number of branches that transmit the same signal (the signal is not divided among these branches in any way). The outgoing branches terminate at the incoming connections of other neurons in the network. Table 2.1 shows the analogy between biological and artificial neural [3].

Table 2.1 Analogy between Biological and Artificial Neural Networks

Biological Neural Network	Artificial Neural Network
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

2.4.1 Components of Neural Network

Typically, neural network requires the important parts to complete its desired tasks, classification, prediction and others. Similar to the brain, a neural network is composed of artificial neurons (or units) and interconnections. The neurons are connected by weighted links passing signals from one neuron to another. Activation functions will be required to obtain successful result.

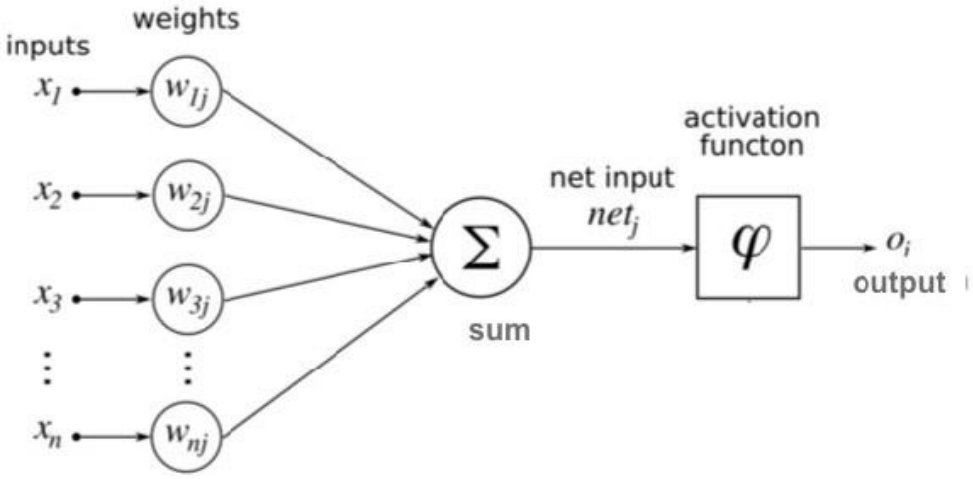

Figure 2.7 Components of Neural Network

2.4.1.1 Neurons

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. Artificial neurons are elementary units in an artificial neural network.

A neuron with label *j* receiving an input $p_j\ (t)$ from predecessor neurons consists of the following components:

- an activation $a_j(t)$, depending on a discrete time parameter,
- possibly a threshold θ_j, which stays fixed unless changed by a learning function,
- an activation function *f* that computes the new activation at a given time $t + 1$ from $a_j(t)$, θ_j and the net input $p_j\ (t)$ giving rise to the relation

$$a_j(t+1) = f\left(a_j(t), p_j(t), \theta_j\right) \tag{2.7}$$

- and an output function f_{out} computing the output from the activation

$$o_j(t) = f_{out}\left(a_j(t)\right) \tag{2.8}$$

An input neuron has no predecessor but serves as input interface for the whole network. Similarly, an output neuron has no successor and thus serves as output interface of the whole network.

2.4.1.2 Connections and Weights

The network consists of connections, each connection transferring the output of a neuron *i* to the input of a neuron *j*. In this sense, *i* is the predecessor of *j* and *j* is the successor of *i*. Each connection is assigned a weight w_{ij}.

2.4.1.3 Activation Function

The activation function computes the input $p_j(t)$ to the neuron *j* from the outputs $o_i(t)$ of predecessor neurons and typically has the form

$$p_j(t) = \sum_i o_i(t)\, w_{ij} \tag{2.9}$$

In neural network, this function is also called the transfer function. It's just a thing (node) that is added to the output end of any neural network. It can also be attached in between two Neural Networks. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

Many activation functions have been tested, but only a few have found practical applications. Four common choices-the step, sign, linear and sigmoid functions- are illustrated in Figure 2.8.

The Activation Functions can be basically divided into 2 types

1) Linear Activation Function and
2) Non-linear Activation Functions.

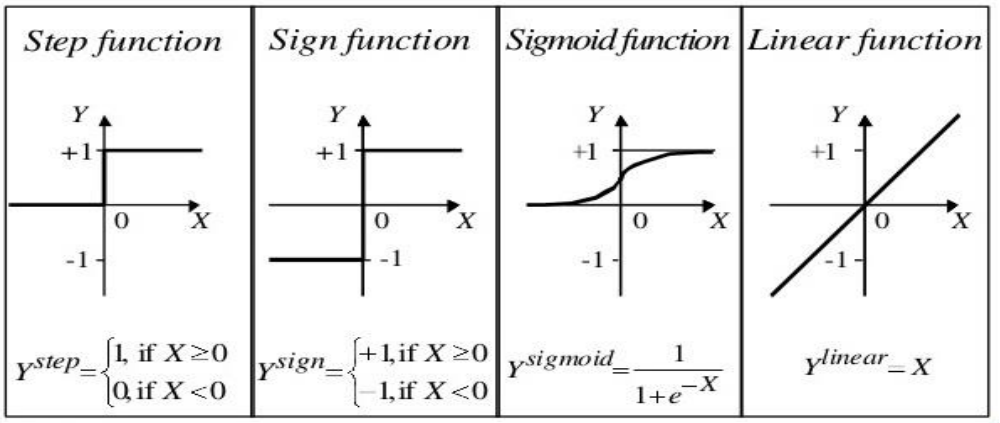

Figure 2.8 Some Activation Functions of Neuron [7]

2.4.1.4 Learning Rule

The learning rule is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This learning process typically amounts to modifying the weights and thresholds of the variables within the network.

2.4.2 Architecture of Neural Network

The pattern of connections between the neurons is generally called the architecture of the neural network. The back-propagation model is one of layered neural networks, since each neural network consists of distinct layers of neurons. There are three layers, called input, hidden, and output layers. Generally, there are one input, one output, and any number of hidden layers. One hidden layer is most common; the next common numbers are zero and two. Three or more hidden layers are very rare [7].

The neurons in the input layer do not compute anything. The number of the neurons in the input and output layers are usually determined from a specific application problem. All the neurons in the input layer are connected to all the neurons in the hidden layers through the edges. Similarly, all the neurons in the hidden layer are connected to all the neurons in the output layer through the edges.

A weight is associated with each connection. Input layer represents the raw information that is fed into the network. This part of network is never changing its values. Every single input to the network is duplicated and sends down to the nodes in hidden layer. Hidden Layer accepts data from the input layer. It uses input values and modifies them using some weight value, this new value is then send to the output layer but it will also be modified by some weight from connection between hidden and output layer.

Output layer process information received from the hidden layer and produces an output layer. Output layer process information received from the hidden layer and produces an output. This output is then processed by activation function.

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

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

2.4.3 Learning Paradigms

The three major learning paradigms each correspond to a particular learning task. These are

- Supervised Learning,
- Unsupervised Learning and
- Reinforcement Learning.

2.4.3.1 Supervised Learning

Supervised learning uses a set of example pairs $(x, y), x \in X, y \in Y$ and the aim is to find a function $f: X \rightarrow Y$ in the allowed class of functions that matches the examples. In other words, in order to infer the mapping implied by the data; the cost function is related to the mismatch between the mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output, $f(x)$, and the target value y over all the example pairs. Minimizing this cost using gradient descent for the class of neural networks called multilayer perceptron (MLP), produces the back-propagation algorithm for training neural networks. The simple process of reinforcement learning is shown in figure 2.9.

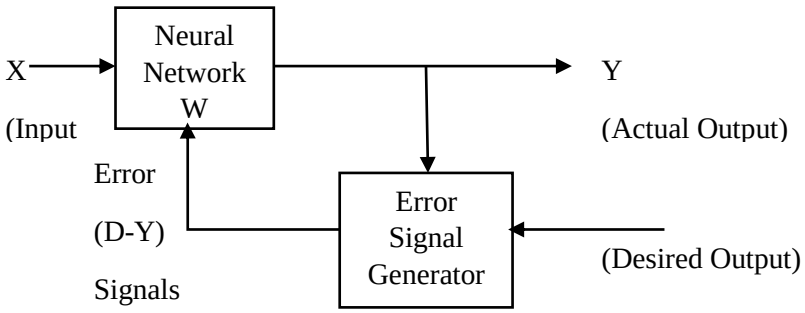

Figure 2.9 Process of Supervised Learning

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function

approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for hand writing, speech and gesture recognition). This can be thought of as learning with a "teacher", in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

2.4.3.2 Unsupervised Learning

In unsupervised learning, some data *x* is given and the cost function to be minimized, that can be any function of the data *x* and the network's output, *f*.

The cost function is dependent on the task (the model domain) and any a priori assumptions (the implicit properties of the model, its parameters and the observed variables).

As a trivial example, consider the model $f(x) = a$ is a constant and the cost $C = E[(x - f(x))^2]$. Minimizing this cost produces a value of *a* that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: for example, in compression it could be related to the mutual information between *x* and *f(x)*, whereas in statistical modeling, it could be related to the posterior probability of the model given the data (note that in both of those examples those quantities would be maximized rather than minimized).

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering. The simple process of reinforcement learning is shown in figure 2.10.

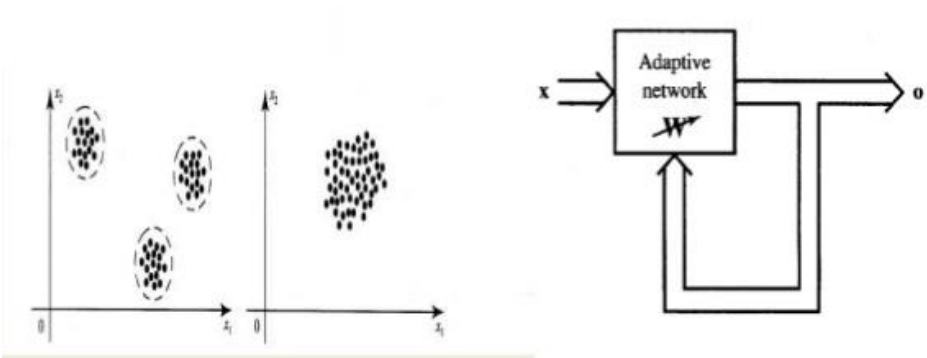

Figure 2.10 Process of Unsupervised Learning

2.4.3.3 Reinforcement Learning

In reinforcement learning, data *x* is usually not given, but generated by an agent's interactions with the environment. At each point in time *t*, the agent performs

an action y_t and the environment generates an observation x_t and an instantaneous cost c_t, according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost, e.g., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally the environment is modeled as a Markov decision process (MDP) with states $s_1, \ldots, s_n \in S$ actions $a_1, \ldots, a_m \in A$ with the following probability distributions: the instantaneous cost distribution $P(c_t \mid s_t)$, the observation distribution $P(x_t \mid s_t)$ and the transition $P(s_{t+1} \mid s_t, a_t)$, while a policy is defined as the conditional distribution over actions given the observations. Taken together, the two then define a Markov chain (MC). The aim is to discover the policy (i.e., the MC) that minimizes the cost.

ANNs are frequently used in reinforcement learning as part of the overall algorithm. Dynamic programming was coupled with ANNs and applied to multi-dimensional nonlinear problems such as those involved in vehicle routing, natural resources management or medicine because of the ability of ANNs to mitigate losses of accuracy even when reducing the discretization grid density for numerically approximating the solution of the original control problems.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks. The simple process of reinforcement learning is shown in figure 2.11.

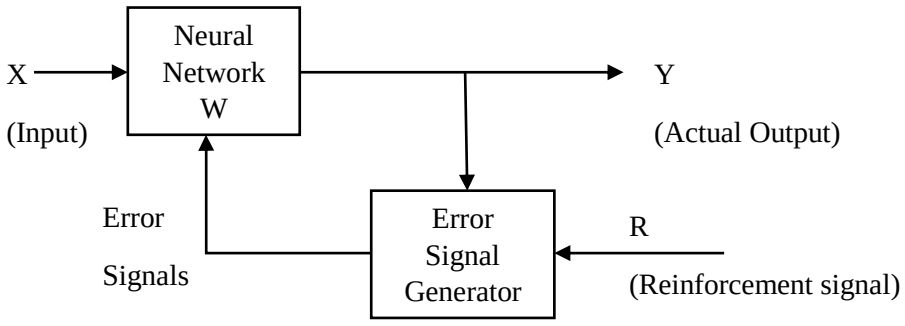

Figure 2.11 Process of Reinforcement Learning

2.4.4 Learning Process

The neurons are connected by links, and each link has a numerical weight associated with it. Weights are the basic means of long-term memory in ANNs. They

express the strength, or in other words importance, of each neuron input. A neural network 'learns' through repeated adjustments of these weights.

In 1943, Warren McCulloch and Walter Pitts proposed a very simple idea that is still the basis for most artificial neural networks. The neuron computes the weighted sum of the input signals and compares the result with a threshold value, θ. If the net input is less than the threshold, the neuron output is -1. But if the net input is greater than or equal to the threshold, the neuron becomes activated and its output attains a value +1. In other words, the neuron uses a transfer or activation function.

The step and sign activation functions, also called hard limit functions, are often used in decision-making neurons for classification and pattern recognition tasks. The sigmoid function transforms the input, which can have any value between plus and minus infinity, into a reasonable value in the range between 0 and 1. Neurons with this function are used in the back-propagation networks. The linear activation function provides an output equal to the neuron weighted input. Neurons with the linear function are often used for linear approximation [3].

2.4.5 Multilayer Neural Network

A multilayer perceptron is a feed-forward neural network with one or more hidden layers. Typically, the multilayer network consists of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer of computational neurons. The input signals are propagated in a forward direction on a layer-by-layer basis.

Each layer in a multilayer neural network has its own specific function. The input layer accepts input signals from the outside world and redistributes these signals to all neurons in the hidden layer. Actually, the input layer rarely includes computing neurons, and thus does not process input patterns. The output layer accepts output signals, or in other words a stimulus pattern, from the hidden layer and establishes the output pattern of the entire network.

Neurons in the hidden layer detect the features of the input patterns. The weights of the neurons represent the features hidden in the input patterns. These features are then used by the output layer in determining the output pattern. With one hidden layer, any continuous function of the input signals can be represented, and with two hidden layers not only continuous function, but also even discontinuous functions can be represented.

More than a hundred different learning algorithms are available, but the most popular method is back-propagation. This method was first proposed in 1969 (Bryson and Ho, 1969), but was ignored because of its demanding computations. Only in the mid-1980s was the back-propagation learning algorithm rediscovered. A training set of input patterns is presented to the network. The network computes its output pattern, and if there is an error - a difference between actual and desired output patterns - the weights are adjusted to reduce this error [3]. The architecture of multilayer neural network is shown as simple diagram in figure 2.12.

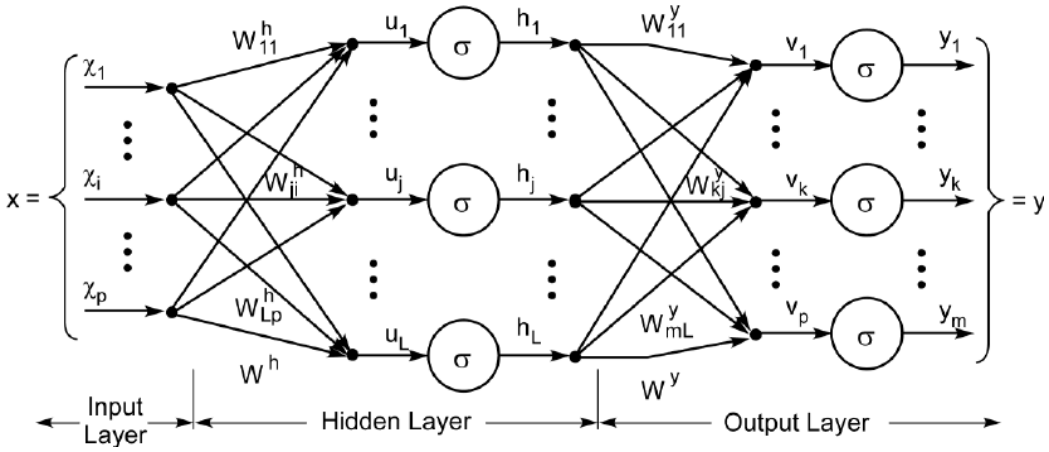

Figure 2.12 Architecture of Multilayer Neural Network [3]

2.4.6 Back-propagation Neural Network

Typically, a back-propagation network is a multilayer network that has three or four layers. The layers are fully connected, that is, every neuron in each layer is connected to every other neuron in the adjacent forward layer. As with any other neural network, a back-propagation one is determined by the connections between neurons (the network's architecture), the activation function used by the neurons, and the learning algorithm (or the learning law) that specifies the procedure for adjusting weights.

In a back-propagation neural network, the learning algorithm has two phases. As the first phase, a training input pattern is presented to the network input layer. As the second phase, the network then propagates the input pattern to generate the output pattern of the hidden layer.

The network then propagates the input pattern which is the output of hidden layer from layer to layer until the output pattern is generated by the output layer. If this pattern is different from the desired output, an error is calculated and then

propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated.

In a simple way, back-propagation neural network is the learning process takes the inputs and the desired outputs and updates its internal state accordingly, so the calculated output get as close as possible from the desired output. The predict process takes input and generate, using the internal state, the most likely output according to its past *"training experience"*.

The back-propagation model is called supervised learning method for this reason, i.e. it learns under supervision. It cannot learn without being given correct sample patterns [3]. The architecture of back-propagation is shown as simple diagram in figure 2.13.

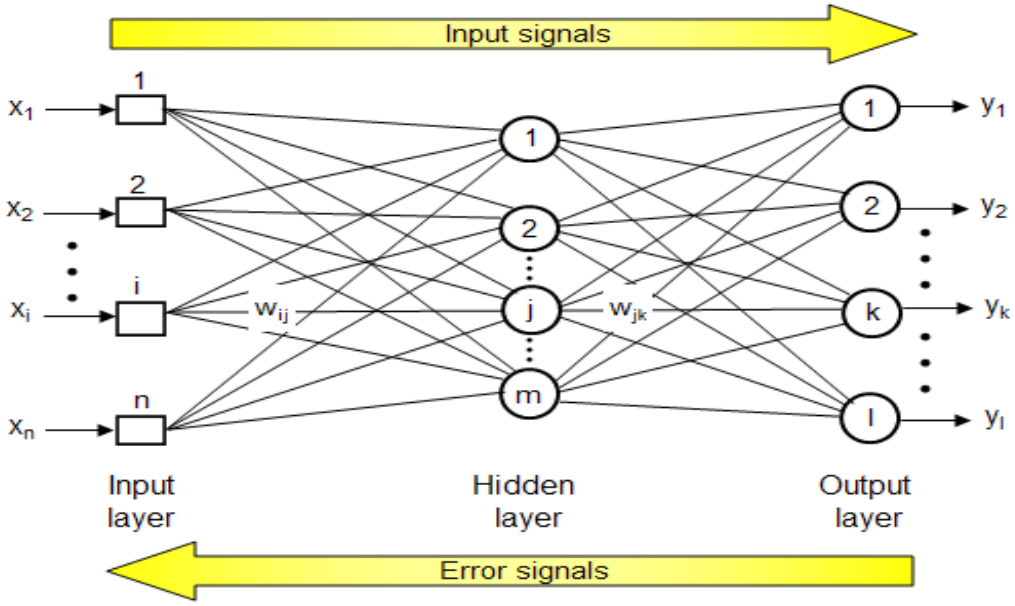

Figure 2.13 Architecture of Back-propagation Neural Network

2.4.6.1 Back-propagation Training Algorithm

Step 1: Initialization

Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range (Haykin, 1999):

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right), \tag{2.10}$$

where Fi is the total number of inputs of neuron i in the network. The weight initialization is done on a neuron-by-neuron basis.

Step 2: Activation

Activate the back-propagation neural network by applying inputs $x_1(p)$, $x_2(p)$, …, $x_n(p)$ and desired outputs $y_{d,1}(p)$, $y_{d,2}(p)$, …, $y_{d,n}(p)$.

(a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j\right], \quad (2.11)$$

where n is the number of inputs of neuron j in the hidden layer, and sigmoid is the sigmoid activation function.

(b) Calculate the actual outputs of the neurons in the output layer:

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k\right], \quad (2.12)$$

where m is the number of inputs of neuron k in the output layer.

Step 3: Weight training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

(a) Calculate the error gradient for the neurons in the output layer:

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p), \quad (2.13)$$

Where

$$e_k(p) = y_{d,k}(p) - y_k(p) \quad (2.14)$$

Calculate the weight corrections:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \quad (2.15)$$

Update the weights at the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \quad (2.16)$$

(b) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = y_i(p) \times [1 - y_i(p)] \times \sum_{k=1}^{j} \delta_k(p) \times w_{jk}(p) \quad (2.17)$$

Calculate the weight corrections:

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \quad (2.18)$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \quad (2.19)$$

Step 4: Iteration

Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied [3].

2.4.6.2 Application Categories of the Back-propagation Model

Three major application categories of the back-propagation model are classification, prediction, and control. These categories are for the convenience of easy understanding, and should not be considered as exhaustive and rigid.

Consider character recognition as an example to determine whether a given pattern is character 'A' or some other character is a simple classification problem. The idea is to recognize and classify given patterns to typically much fewer groups of patterns. The latter will be the output of the network for this type of applications. This is accomplished by training the neural network using sample patterns and their correct answers. When the neural network is properly trained, it can give correct answers for not only the sample patterns, but also for new similar patterns. The input can be in variety of forms, not just a visual image.

When using neural networks for prediction problems, time must be incorporated as one factor. For example, suppose that a neural network is given many patterns under different circumstances over a certain length of time. Given a similar pattern in its earlier stage, the neural network may be able to predict the most likely pattern to follow. Vital issues are whether the appropriate factors are included in the model and whether they are accurately measured. For example, describing the correct financial model to predict the stock market would be difficult.

The control problem can be considered as a mapping problem from input, which may include feed-in attributes and possible feedback, to output of control parameters. The mappings of different input-to-output values can be viewed as patterns, and they can be learned by a neural network. For example, consider controlling a steel furnace. Inputs are various physical and chemical measurement distributions, such as temperature, pressure, and various chemical components at different points within the furnace.

Outputs are the quantities to be supplied for the heat source, such as coal, gas and air, raw material, and so forth. Many patterns representing various input-to-output

mappings can be learned by the neural network; then it can be used to control the furnace. This is an example of plant control. The same basic concept can be applied to various components of transportation equipment such as an airplane and car, robots, and so on [7].

2.4.7 Advantages and Disadvantages of Neural Network

One major advantage of neural networks is that their easy implementation of parallelism since, for example, each neuron can work independently. Other advantages often cited include:

Robustness: For example, neural networks can deal with certain amount of noise in the input. Even if part of a neural network is damaged (perhaps similar to partial brain damage), often it can still perform tasks to a certain extent, unlike some engineering systems, like a computer.

Generalization: A neural network can deal with new patterns which are similar to learned patterns.

Nonlinearity: Nonlinear problems are hard to solve mathematically. Neural networks can deal with any problems that can be represented as patterns.

The disadvantages of neural networks include the following:

- First, they have not been able to mimic the human brain or intelligence. They just only can learn trained information.
- Second, after a neural network is successfully trained to perform its goal, its weights have no direct meaning to us. That is, any underlying rules which may be implied from the neural network cannot be extracted.
- Third, computation often takes a long time, and sometimes it does not even converge. A counter-argument against this common problem of long time training is that even though it may take a month of continuous training, once it is successful, it can be copied to other systems easily and the benefit can be significant.
- Fourth, scaling up a neural network is not a simple matter. For example, suppose that a neural network was trained for 100 input neurons. When the trained network need to be extracted to a neural network of 101 input neurons, normally it is necessary to start over an entire training session for the new network [7].

2.5 Android Software Development

Android software development is the process by which new applications are created for devices running the Android operating system. Android apps can be written using Kotlin, Java, and C++ languages using the Android software development kit (SDK), while using other languages is also possible.

All non-JVM languages, such as Go (JavaScript, C, C++ or assembly), need the help of JVM language code, which may be supplied by tools, likely with restricted API support. Some languages/programming tools allow cross-platform app support, i.e. for both Android and iOS.

2.5.1 Android SDK

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android OS

The development platforms that are compatible with SDK include operating systems like Windows (XP or later), Linux (any recent Linux distribution) and Mac OS X (10.4.9 or later). The components of Android SDK can be downloaded separately. Third party add-ons are also available for download.

Although the SDK can be used to write Android programs in the command prompt, the most common method is by using an integrated development environment (IDE). The recommended IDE is Eclipse with the Android Development Tools (ADT) plug-in. Android studio IDE is also widely used. The android SDK tools compile the code along with any data and resource files into an android package.

However, other IDEs, such as NetBeans or IntelliJ, will also work. Most of these IDEs provide a graphical interface enabling developers to perform development tasks faster. Since Android applications are written in Java code, a user should have the Java Development Kit (JDK) installed [10].

2.5.2 Android Library

An Android library is structurally the same as an Android app module. It can include everything needed to build an app, including source code, resource files, and an Android manifest. However, instead of compiling into an APK that runs on a device, an Android library compiles into an Android Archive (AAR) file that can be used as a dependency for an Android app module. Unlike JAR files, AAR files can contain Android resources and a manifest file, which allows users to bundle in shared resources like layouts and drawables in addition to Java classes and methods.

A library module is useful in the following situations such as:

- Building multiple apps that use some of the same components, such as activities, services, or UI layouts.
- Building an app that exists in multiple APK variations, such as a free and paid version and needs the same core components in both.

In either case, simply move the files that are necessary to reuse into a library module and then add the library as a dependency for each app module [11].

2.5.3 Android Signature Pad

Android Signature Pad is an Android library for drawing smooth signatures. It uses variable width Bezier curve interpolation based on Smoother Signatures.

Features of android Signature Pad library are as follow:

- Bezier implementation for a smoother line
- Variable point size based on velocity
- Customizable pen color and size
- Bitmap and SVG support
- Data Binding

CHAPTER 3
SYSTEM DESIGN AND IMPLEMENTATION

3.1 System Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

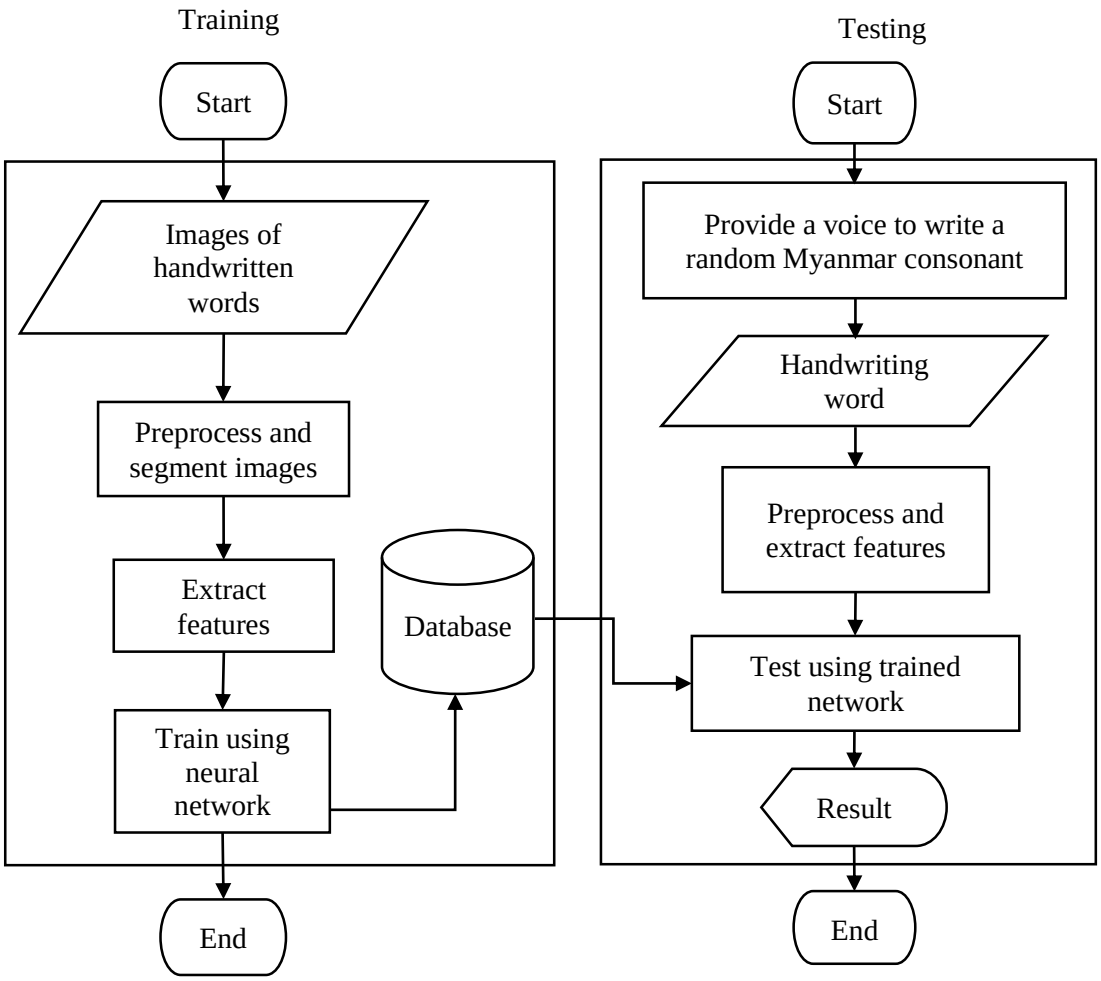

Figure 3.1 System Design

The design diagram for the system is shown in figure 3.1. The system has two parts. First part is training. In the training part, the images of handwritten words are acquired. The acquired images are preprocessed and segmented. Then the features of segmented images are extracted. Finally, the extracted features are trained with neural network. Second part is testing. In the testing part, the image from android storage is acquired. Then the acquired image is preprocessed and extracted features and test with the trained network. Then the result is shown in android.

3.2 System Flow

Flow diagram is a collective term for a diagram representing a flow or set of dynamic relationships in a system. The term flow diagram is also used as a synonym for flowchart, and sometimes as a counterpart of the flowchart. Flow diagrams are used to structure and order a complex system, or to reveal the underlying structure of the elements and their interaction. It is a diagram that visually displays interrelated information such as events, steps, in a process, functions, etc., in an organized fashion, such as sequentially or chronologically.

Figure 3.2 is the system flow of the system. As the system is handwriting recognition system, the system involves two mainly functions of studying and practicing. Users can study the writing and pronunciation of the consonant they want to know from all of the thirty three consonants. Users can practice writing and listening of the consonants using this application. The system can automatically check whether the users thoroughly know the pronunciation and the writing of the consonants.

3.3 System Implementation

The system is an implementation of an application for kids and illiterates to learn thirty three Myanmar consonants. The system is implemented in two versions. For the users who want to check the written consonants are similar to what consonant exactly, the users can input an image of written consonant and test in desktop program. The system is also implemented as an android application. The application allows the user to learn the pronunciation and writing of the consonants.

The system is developed using matlab programming and android programming. The system uses firebase storage to store the image of the written word and associated text file. The system also use Php code to share result data between matlab and android program.

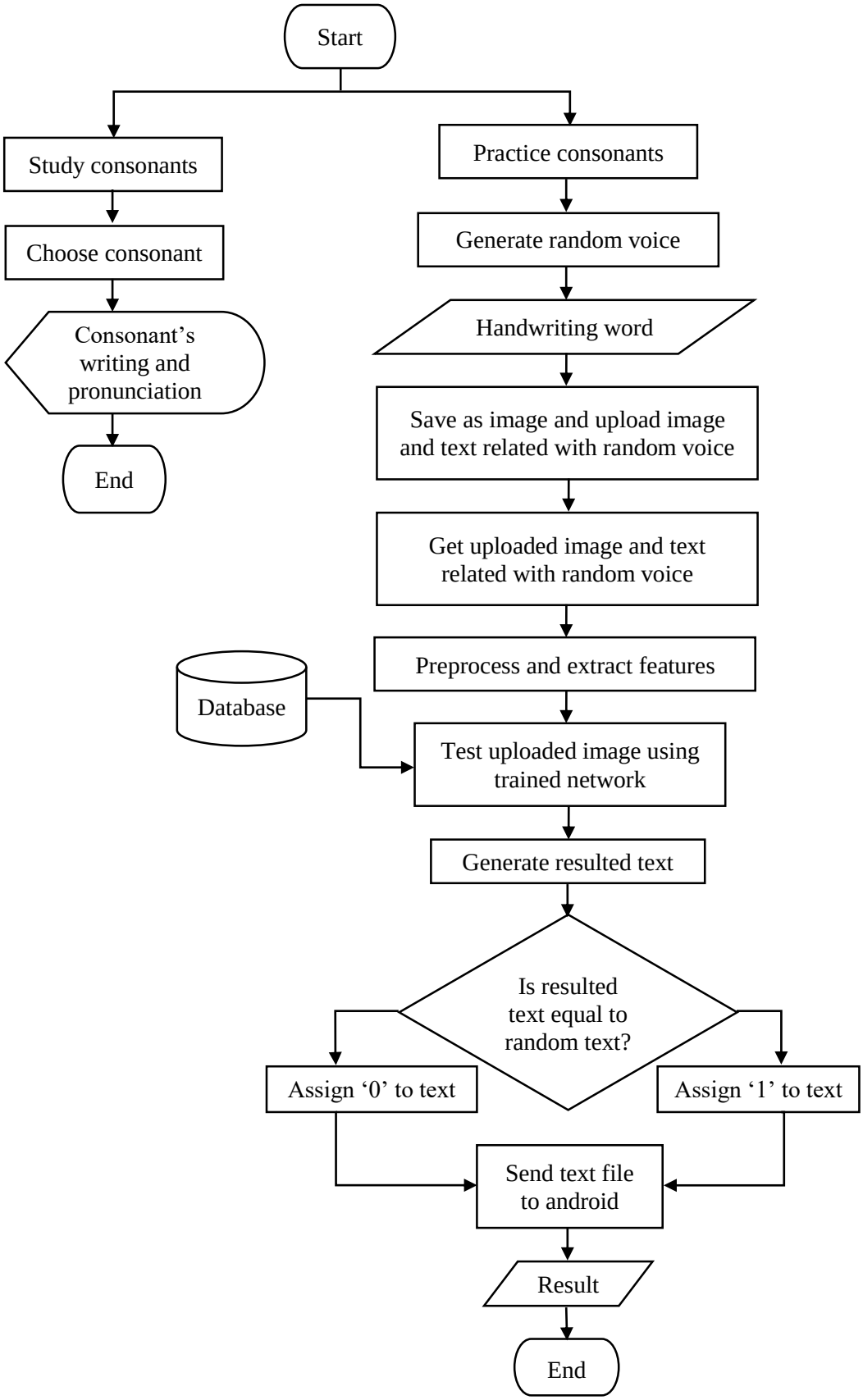

Figure 3.2 System Flow

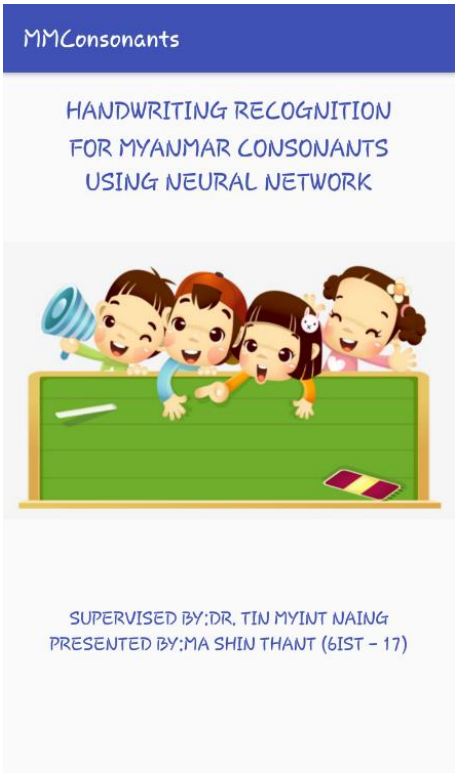

Figure 3.3 Welcome Screen

Figure 3.3 is the welcome screen of the system. After showing this welcome screen about three seconds, home page will appear.

Figure 3.4 Home Page

Figure 3.4 is the home page of the system. There are two buttons in this page. Study button will lead to the study page and Practice button will lead to the practice page.

The main study page is shown in figure 3.5. This is the study page where users can select any consonant they desired to study by clicking the button of the desired consonant. In this page, the users can study the overview writing of all consonants. The button with home icon will lead to home page, the previous page.

Figure 3.6 is the study page in which users can study the way of writing of the chosen consonant. The back button at the upper left corner will lead to the main study page, the previous page. The button at the lower left corner will lead to the page with the next consonant in descending order. The button at the lower middle will provide the pronunciation of the consonant. The button at the lower right corner will lead to the page with the next consonant in ascending order.

The page for users to practice is shown in figure 3.7.The practice page in which the users need to write according to the random pronunciation. The home button will lead to the home page. The button at the lower left corner will clear the screen. The button at the lower right corner will save the written word. The button at the upper left corner will provide the practice page again with another random pronunciation.

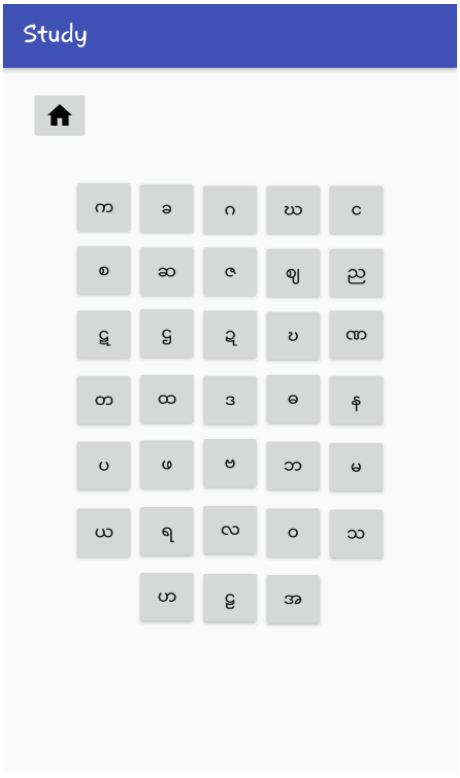

Figure 3.5 Main Study Page

Figure 3.6 Study Page

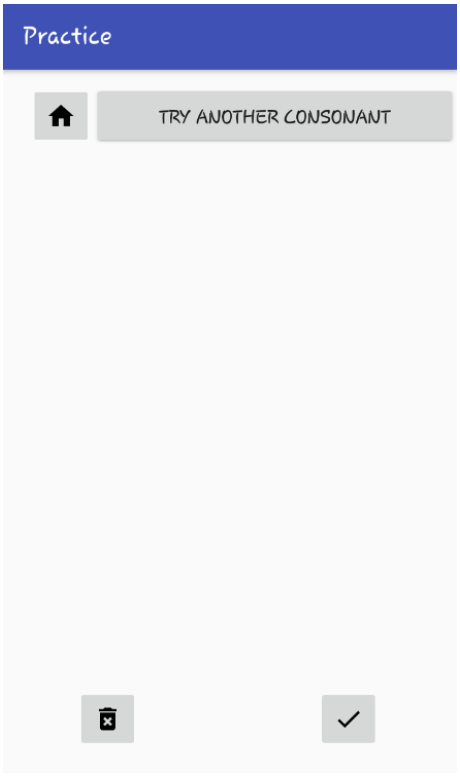

Figure 3.7 Practice Page

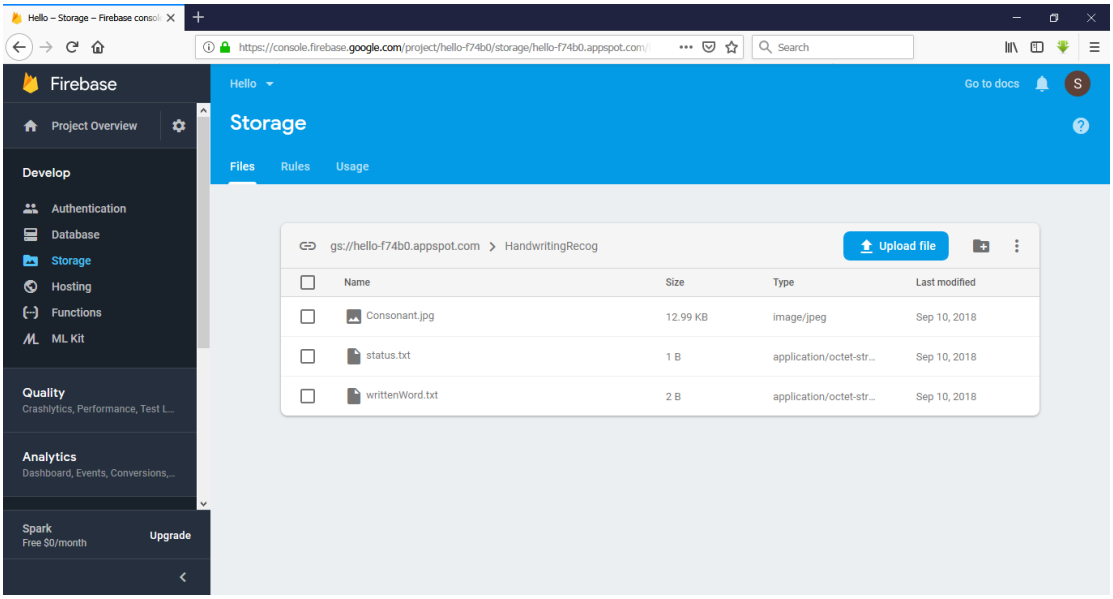

Figure 3.8 Data Stored in Firebase Storage

The console of firebase with stored image and associated text file is shown in figure 3.8. The written consonant is saved as image and is uploaded to firebase. Status file is also uploaded with one digit '1' in order to notify the matlab that the image is uploaded. Another file is also uploaded with the random number that represents the random pronunciation in order to check the written word is correct or not.

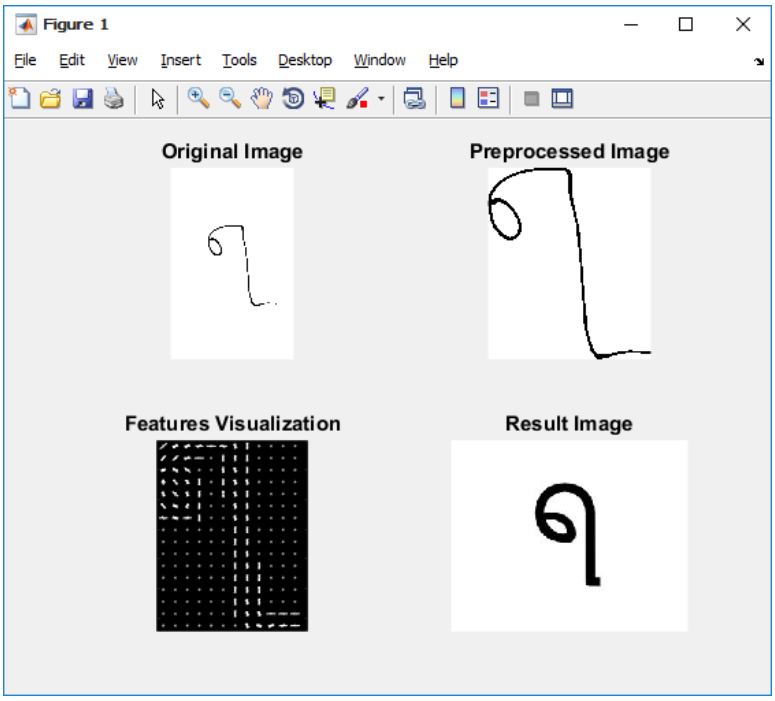

Figure 3.9 Classification of the Acquired Image

The process that matlab acquires the uploaded files using links generated from firebase is shown in figure 3.9. Then matlab performs preprocessing, features extraction and testing steps to the acquired image and the network returns a value that is recognized.

Figure 3.10 shows the web page written in Php. This php page is called by the matlab program after the neural network recognizes the written word. This php page is used to note the network return value. If the recognized value is equal to the random number in the acquired file, result number '1' is written in the result file. Otherwise, result number '0' is written. Then matlab acquires a php file which send the value in the result file to android studio.

When the user click the save button, figure 3.11 and figure 3.12 will appear. Figure 3.11 is the page that appeared when the written consonant is correct. If the written consonant is correct, then the message 'Correct Answer' will show. Clapping voice and happy emoji will also appear to notify that the consonant is correct to kids who are unable to read the message.

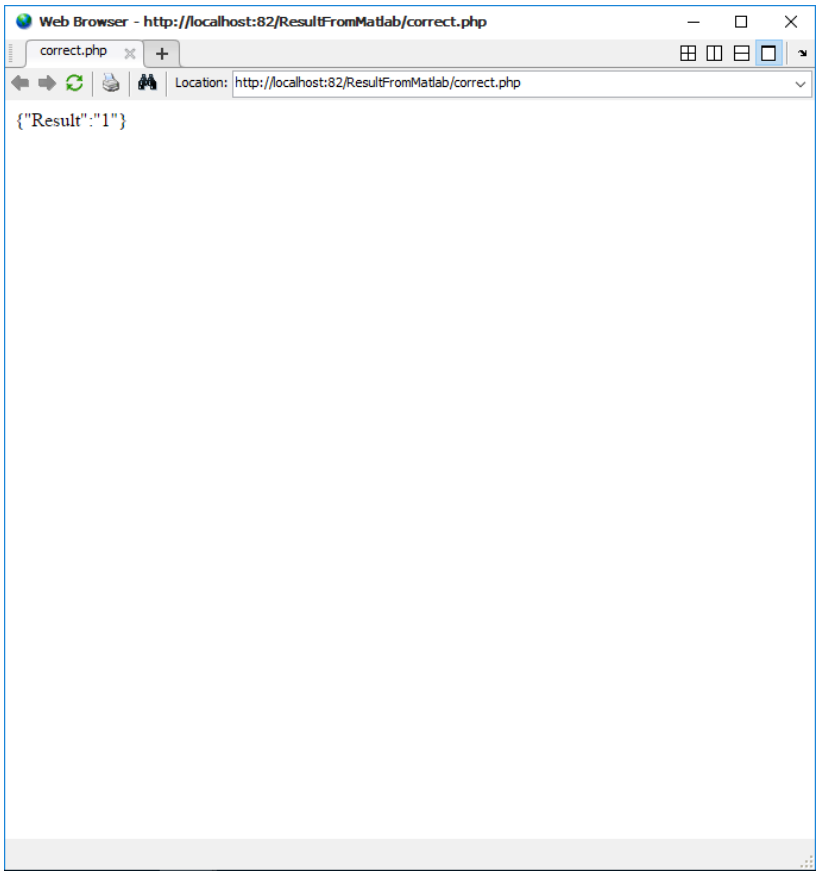

Figure 3.10 Web Page in Php

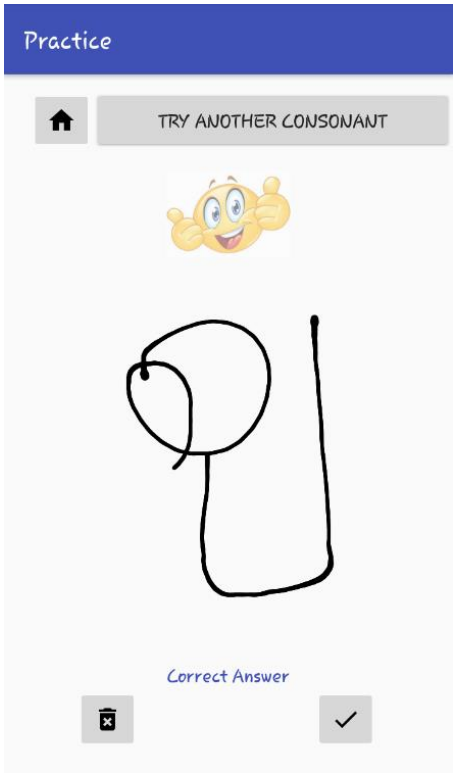

Figure 3.11 Practice Page with Correct Result

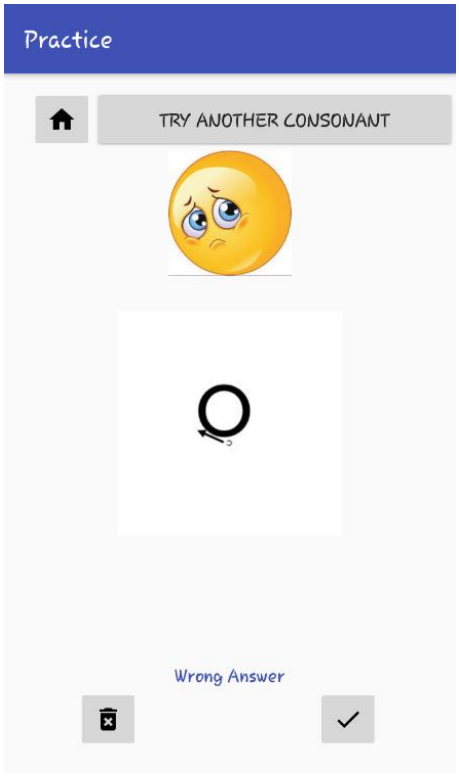

Figure 3.12 Practice Page with Wrong Result

Figure 3.11 is the page that appeared when the written consonant is correct If the written consonant is wrong, then the message ‘Wrong Answer’ will show together with the correct consonant. Failure voice and sad emoji will also appear to notify that the consonant is correct to kids who are unable to read the message.

Figure 3.13 Exit Page

Figure 3.13 is an exit page with an alert that asks users to exit or not that appears when users click the back button of phone in home page.

Figure 3.14 Home Page in Matlab

Figure 3.14 is the home page in matlab program. The system is also implemented to test using existing consonants in desktop. Clicking the start button will lead to figure 3.15.

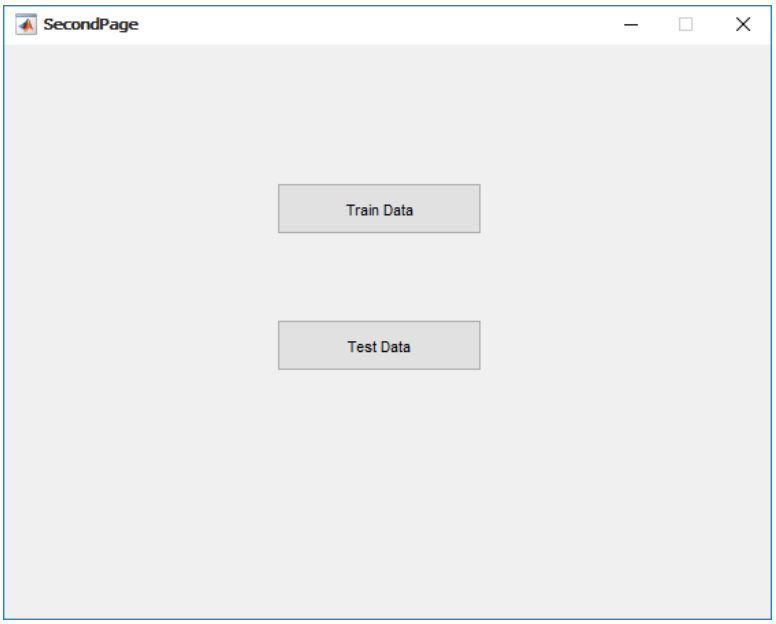

Figure 3.15 Second Page in Matlab

Figure 3.15 is the page that will appear after main page. In this page, users can train the consonants using neural network by clicking the 'Train Data' button. In order to test, users need to click 'Test Data' button.

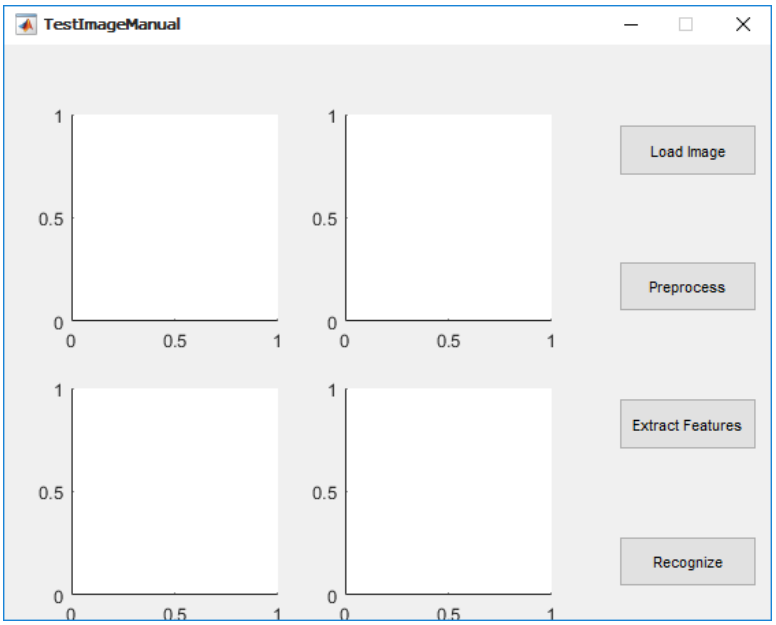

Figure 3.16 Testing Page in Matlab

As shown in figure 3.16, users can load the written consonant image, then preprocess, extract the features of the preprocessed image and then recognize what the written consonant is.

Figure 3.17 shows an example of written consonant image and how the system works in desktop.

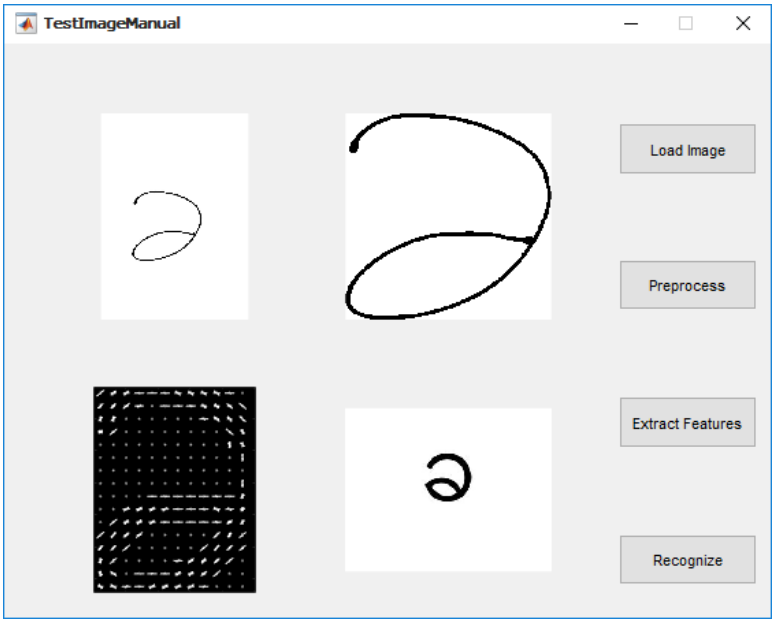

Figure 3.17 Testing Page with Tested Data in Matlab

3.4 Accuracy of the System

Table 3.1 Confusion Matrix for the Accuracy of the System

	က	ခ	ဂ	ဃ	င	စ	ဆ	ဇ	ဈ	ည	ဋ	ဌ	ဍ	ဎ	တ	ထ	ဒ	ဓ	န	ပ	ဖ	ဗ	ဘ	မ	ယ	ရ	လ	ဝ	သ	ဟ	ဠ	အ	Total
က	7															1												2						10
ခ		10																															10
ဂ			9																									1					10
ဃ				10																													10
င					8																							2					10
စ						10																											10
ဆ							9									1																	10
ဇ								10																									10
ဈ									10																								10

ည										10																					10
ဋ											10																				10
ဌ												10																			10
ဍ													10																		10
ဎ														10																	10
ဏ															10																10
တ																10															10
ထ																	8							1		1					10
ဒ																		10													10
ဓ																			6					2			2				10
န																				10											10
ပ																					10										10
ဖ																						9			1						10
ဗ																							10								10
ဘ															3									7							10
မ																									10						10
ယ																										10					10
ရ																											10				10
လ																										1		9			10
ဝ																													10		10
သ																	2												8		10
ဟ																	1								2					7	10
ဠ																														10	10
အ																								1							9	10
Total	7	10	9	10	8	10	9	10	10	10	10	10	10	10	10	14	12	10	6	10	10	9	10	8	12	15	10	12	15	8	7	10	9	330

$$Accuracy = \frac{The\ total\ number\ of\ correct\ answers}{The\ total\ number\ of\ answers} \quad (3.1)$$

$$Accuracy\ of\ the\ system = \frac{306}{330} = 0.9273$$

Therefore, the accuracy of the system is approximately 92%.

3.5 Output of the System

The system would be able to help kids for convenient learning environment. It would provide the benefits of handwriting such as increasing learning comprehension, calming the body and nerves and enhancing focus, etc.

In this chapter, the system design, system flow diagram, the system implementations are expressed. In addition, detail implementation and GUI interface are also included in this chapter.

CHAPTER 4
CONCLUSION

Education is vital field in human society. Technology can provide more comfortable educational zone. There are a lot of things developed using educational technology in many different ways. Mobile applications are also part of educational technology. Users can learn and practice many kinds of languages, knowledge using mobile applications.

The most basic part to learn the reading and writing of a language is to know thoroughly the consonants of that language. This system is about learning thirty three Myanmar consonants. The system can provide the users to learn the way of writing and pronunciation of each consonants. The system can also provide the users to practice random consonants.

The recognition process is performed using matlab programming. The written consonant is saved as image. The image is preprocessed and extracted features using histogram of oriented gradients (HOG). Then the extracted features are tested with the trained network. The system would be able to help kids for convenient learning environment. It would provide the benefits of handwriting such as increasing learning comprehension, calming the body and nerves and enhancing focus, etc.

4.1 Advantages of the System

There are many advantages using this system.

- The users can learn how to write Myanmar consonants.
- The users can also learn the pronunciation of Myanmar consonants.
- The users can reduce requirements such as paper and pencil to practice.
- The users can practice and know the consonant they write is correct or not without the need of a teacher.

4.2 Limitation of the System

There are some limitations using this system.

- The system cannot allow to practice desired consonant.
- The system still need internet to connect to the matlab program.

- The system involves only thirty-three Myanmar consonants.

4.3 Further Extension

In the future, this system can be implemented as a pure android application using opencv (Open Source Computer Vision) library. Scores calculation can also be added to this system. Other related educational tests can also be additional functions of this system.

REFERENCES

[1] Bing Quan Huang, Y.B. Zhang and M.T.Kechadi, “Preprocessing Techniques for Online Handwriting Recognition”, 2007.

[2] Chris Solomon and Toby Breckon, “Fundamentals of Digital Image Processing”, 2011.

[3] Michael Negnevitsky, “Artificial Intelligence: A Guide to Intelligent Systems”, 2011.

[4] Nafiz Imtiaz Bin Hamid, “Projection based Feature Extraction Process for Bangla Script: A modified Approach”, Conference Paper July 2009.

[5] Naveet Dalal and Bill Triggs, “Histogram of Oriented Gradients for Human Detection”, 2005.

[6] School of Computer Science and Statistics, Trinity College and Dublin, Ireland, “Histogram of Oriented Gradients”.

[7] Toshinori Munakata, “Fundamentals of the New Artificial Intelligence”, 2008.

[8] Yi Xiao Yun, “Analysis and Classification of Objects Poses”, 2011.

[9] http://www.wikiwand.com/en/Handwritingrecognition

[10] https://www.techopedia.com/definition/4220/android-sdk

[11] https://developer.android.com/studio/projects/android-library

[12] https://github.com/gcacace/android-signaturepad

[13] https://medium.com/bbm406f17/week3-predicting-the-location-of-o-photograh-8c436e1785f

[14] https://www.researchgate.net/figure/Image-Preprocessing-before-calculating-HOG-features-is-necessary-to-pre-process-the_fig1_33422796