BHARATHI – A HANDWRITTEN TAMIL CHARACTER RECOGNIZER FOR ANDROID

A PROJECT REPORT

Submitted by

ARAVIND.C	**21609104011**
JAGADEESAN.K	**21609104031**
MAHARAJAN.A	**21609104036**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SAVEETHA ENGINEERING COLLEGE, THANDALAM

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2013

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"BHARATHI – A HANDWRITTEN TAMIL CHARACTER RECOGNIZER FOR ANDROID"** is the bonafide work of **"ARAVIND.C (21609104011), JAGADEESAN.K (21609104031)** and **MAHARAJAN.A (21609104036)"** who carried out the project work under my supervision.

SIGNATURE

Mr. R. SARAVANAN, M.E.,(Ph.D.),
Associate Professor

HEAD OF THE DEPARTMENT

Dept. of Computer Science and Engineering,
Saveetha Engineering College,
Thandalam, Chennai 602105.

SIGNATURE

Mr. G. NAGAPPAN, M.E.,(Ph.D.),
Associate Professor

SUPERVISOR

Dept. of Computer Science and Engineering,
Saveetha Engineering College,
Thandalam, Chennai 602105.

DATE OF THE VIVA VOCE EXAMINATION:……..…

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our honourable and beloved Founder President ***Dr. N. M. Veeraiyan***, our President ***Dr. Saveetha Rajesh,*** our Director ***Dr. S. Rajesh*** and other management members for providing the infrastructure needed.

We express our gratitude to our principal ***Dr. R. Venkatasamy, B.Sc., B.Tech., M.E., Ph.D., M.I.E., M.I.S.T.E.,*** and ***Prof. R. Dheenadayalu,*** Dean (ICT) for their whole-hearted encouragement in completing this project.

We convey our thanks to ***Mr. R. Saravanan, M.E., (Ph.D.),*** Associate Professor and Head of Department of Computer Science and Engineering, Saveetha Engineering College, for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks and deep sense of gratitude to our supervisor ***Mr. G. Nagappan, M.E., (Ph.D.),*** Associate Professor, Department of Computer Science and Engineering, Saveetha Engineering College, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of the project work.

We express our thanks to the project coordinator ***Mr. V. Perumal***, ***M.E., (Ph.D.),*** Associate Professor, Department of Computer Science and Engineering, for providing us necessary support and details at the right time and during the progressive reviews.

We owe our thanks to all the members of our college, ***faculty, staff*** and ***technicians*** for their kind and valuable co-operation during the course of the project.

We are pleased to acknowledge our sincere thanks to ***our beloved parents, friends and well wishers*** who encouraged us to complete this project successfully.

ABSTRACT

This project "**BHARATHI**" is aimed at developing software utility which will recognise handwritten characters of Tamil language script and can be accessed through an Input Method Editor.

With rising touch enabled smart-phones and tablet market in India, there is strong need to develop software that provides native language support. This utility enables native users to overcome language barrier in access to technology, by recognizing Tamil characters and numerals. It is also helpful in recognizing special symbols. It engulfs the concept of neural network.

One of the primary means by which computers are endowed with human-like abilities is through the use of a neural network. Neural networks are particularly useful for solving problems that cannot be expressed as a series of steps, such as recognizing patterns, classifying them into groups, series prediction and data mining. The neural network which is trained for classification is designed to take input samples of a hand written data pattern, which then attempts to classify them into groups to determine if the input data matches a pattern that it has memorized.

This project is targeted on Android based Aakash tablets and concerns detecting free handwritten characters through touch gestures. It can be further developed to recognize the characters of different languages.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADT	Android Developer Tools
SDK	Software Development Kit
DVM	Dalvik Virtual Machine
GUI	Graphical User Interface
HWR	Hand Writing Recognition
ANN	Artificial Neural Networks
SOM	Self Organizing Maps
I/O	Input and Output
PE	Processing Elements
KNN	Kohonen Neural Networks
UML	Unified Modelling Language
IMF	Input Method Framework

CHAPTER 1

INTRODUCTION

1.1 AIM

To develop "Bharathi – A Handwritten Tamil Character Recognizer For Android", an Input method editor which uses pattern recognition techniques to recognize handwritten strokes in Tamil on the touch screen. It will eliminate the difficulties faced by Aakash users in using Tamil language soft keyboards presently available in the market.

1.2 OVERVIEW OF THE PROJECT

There is an extensive usage of paper and handwriting in all parts of the world and there is poor penetration of personal computers and keyboard devices. There are only 15 persons per 1000 people who have access to personal computers in India, as compared to 762 in USA, and over 500 in most European countries. With this scenario, touch based interfaces may play an important role in reaping the benefits of Information Technology.

With fast growing market for touch enabled smart-phones and tablets in India, there is strong need to develop software that provides native language support. This utility enables native users to overcome language barrier in access to technology, by recognizing Indic handwritten character inputs. This project is particularly targeted to Aakash due to its immense potential in reaching the Indians.

1.3 ARTIFICIAL NEURAL NETWORKS

Bharathi at its core uses Artificial neural network for pattern recognition, in particular Self Organizing Maps(SOM), which will be briefed in later sections. An artificial neural network, often just named a neural network, is a mathematical model inspired by biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases a neural network is an adaptive system changing its structure during a learning phase. Neural networks are

used for modeling complex relationships between inputs and outputs or to find patterns in data.

1.3.1 KOHONEN NEURAL NETWORK

Kohonen Self Organising Feature Maps, or SOMs. They were invented by a man named, Teuvo Kohonen, a professor of the Academy of Finland, and they provide a way of representing multidimensional data in much lower dimensional spaces - usually one or two dimensions. This process, of reducing the dimensionality of vectors, is essentially a data compression technique known as vector quantization. In addition, the Kohonen technique creates a network that stores information in such a way that any topological relationships within the training set are maintained.A common example used to help teach the principals behind SOMs is the mapping of colours from their three dimensional components - red, green and blue, into two dimensions.

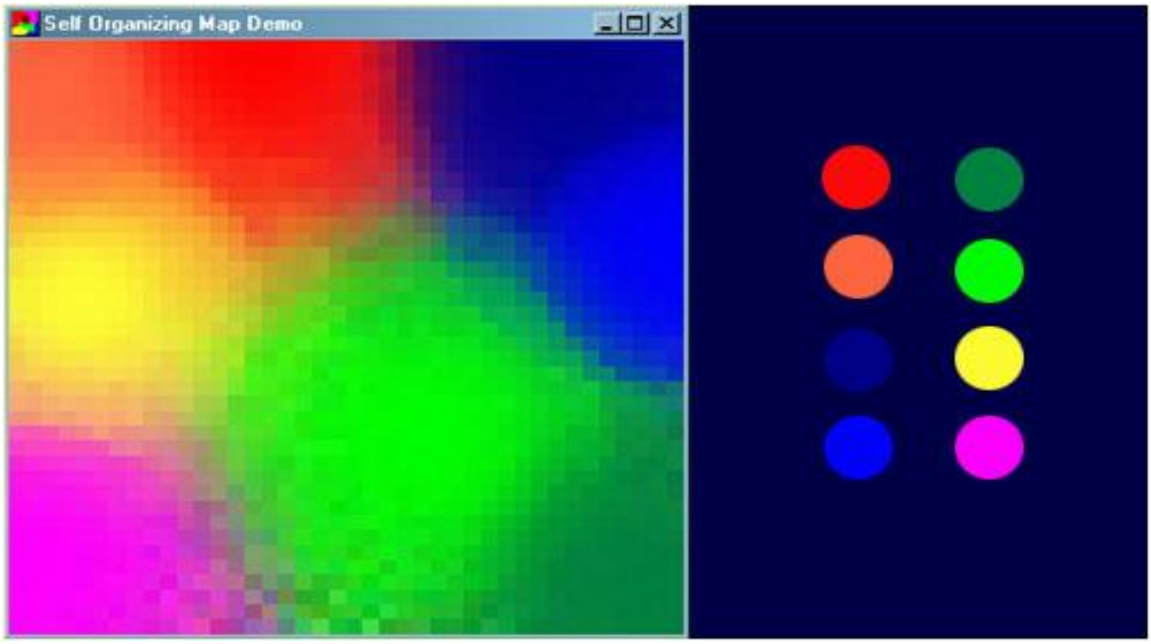


FIG. NO. 1.1 : SELF ORGANINZING MAPS DEMO

The above figure shows an example of a SOM trained to recognize the eight different colours shown on the right. The colours have been presented to the network as 3D vectors - one dimension for each of the colour components - and the network has learnt to represent them in the 2D

space you can see. Notice that in addition to clustering the colours into distinct regions, regions of similar properties are usually found adjacent to each other.

1.3.2 HOW A KOHONEN NETWORK LEARNS

There several steps involved in this training process. Overall the process for training a Kohonen neural network involves stepping through several epochs until the error of the Kohonen neural network is below acceptable level. In this section we will learn these individual processes.

The training process for the Kohonen neural network is competitive. For each training set one neuron will "win". This winning neuron will have its weight adjusted so that it will react even more strongly to the input the next time. As different neurons win for different patterns, their ability to recognize that particular pattern will be increased.

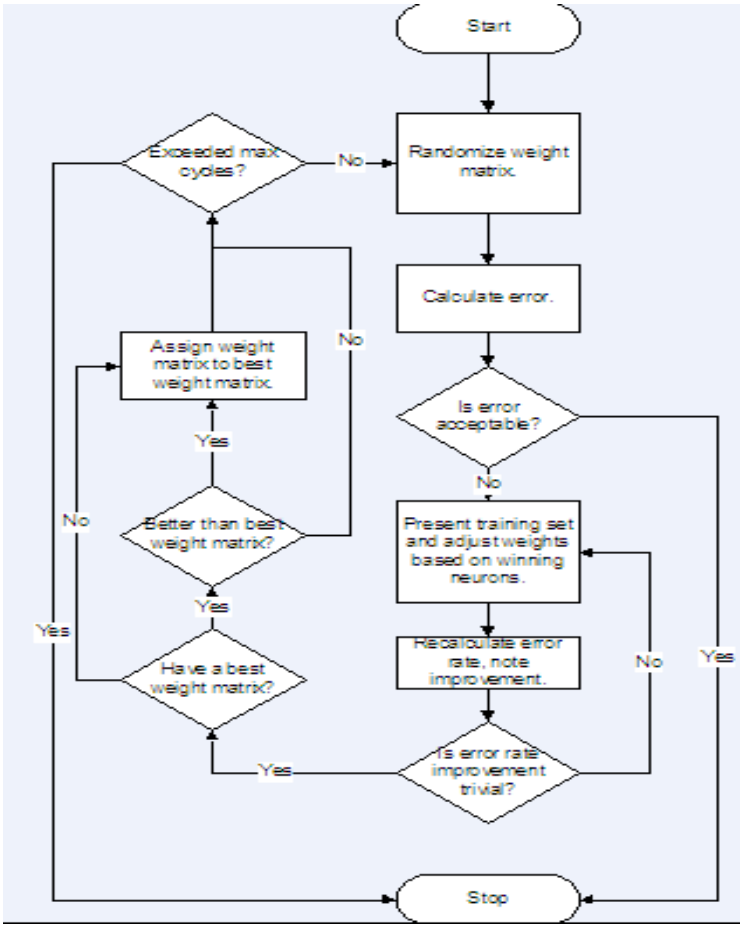


FIG. NO. 1.2: TRAINING THE KOHONEN NEURAL NETWORK

From the above diagram to Kohonen neural network is trained by repeating epochs until one of two things happens. If they calculated error is below acceptable level business at block will complete the training process. On the other hand, if the error rate has all only changed by a very marginal amount this individual cycle will be aborted with tile any additional epochs taking place. If it is determined that the cycle is to be aborted the weights will be initialized random values and a new training cycle began. This training cycle will continue the previous training cycle and that it will analyze epochs on to solve get the two is either abandoned or produces a set of weights that produces an acceptable error level.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

There are number of Input method editors for Android and other desktop application available in the market, this section attempts to brief some of it which are in Input Method Editor category.

2.2 ON-SCREEN INPUT METHODS

Starting from Android 1.5, the Android platform offers an Input Method Framework (IMF) that lets you create on-screen input methods such as software keyboards. This section provides an overview of what Android input method editors (IMEs) are and what an application needs to do to work well with them. The IMF is designed to support new classes of Android devices, such as those without hardware keyboards, so it is important that your application works well with the IMF and offers a great experience for users.

2.2.1 WHAT IS AN INPUT METHOD?

The Android IMF is designed to support a variety of IMEs, including soft keyboard, hand-writing recognizers, and hard keyboard translators. Our focus, however, will be on soft keyboards, since this is the kind of input method that is currently part of the platform.

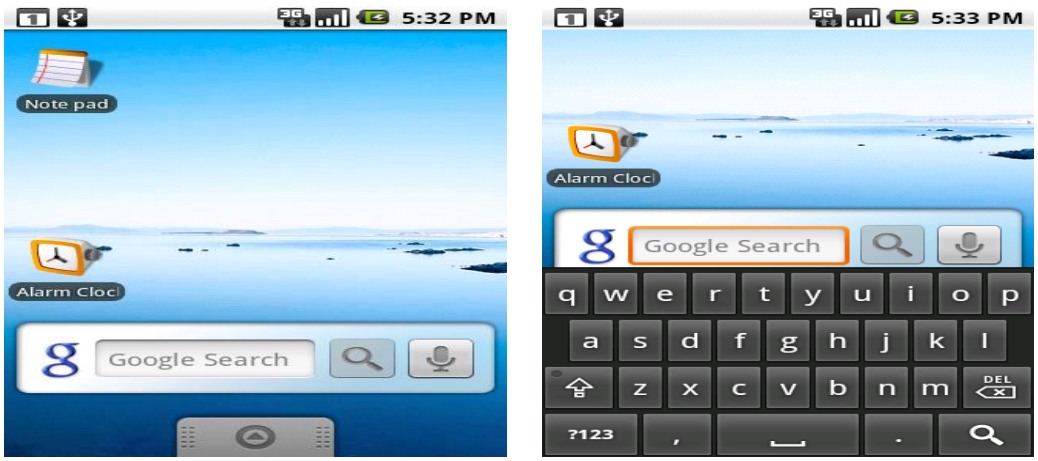


FIG. NO. 2.1: SAMPLE ANDROID HOME SCREEN

A user will usually access the current IME by tapping on a text view to edit, as shown above in the home screen.

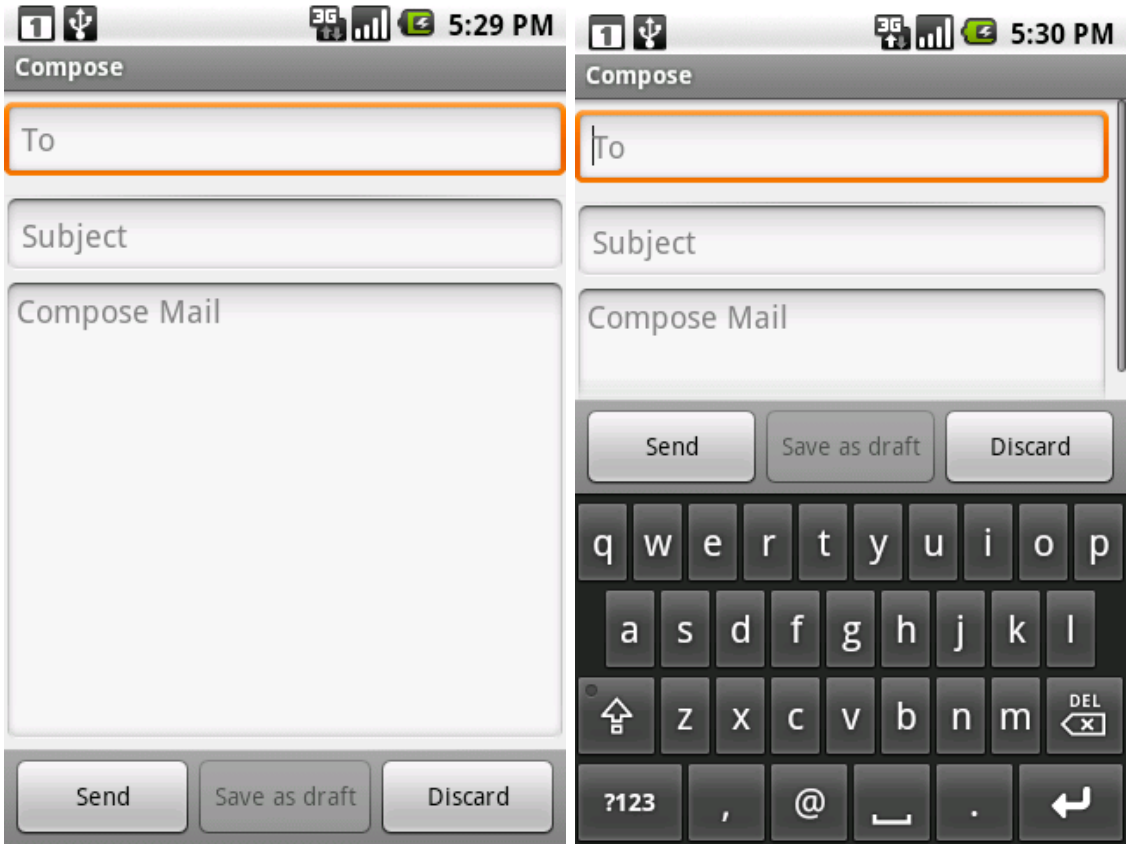


FIG. NO. 2.2 : ACCESSING SOFT KEYBOARD

The soft keyboard is positioned at the bottom of the screen over the application's window. To organize the available space between the application and IME, we use a few approaches; the one shown here is called pan and scan, and simply involves scrolling the application window around so that the currently focused view is visible. This is the default mode, since it is the safest for existing applications.

Most often the preferred screen layout is a resize, where the application's window is resized to be entirely visible. An example is shown here, when composing an e-mail message:

The size of the application window is changed so that none of it is hidden by the IME, allowing full access to both the application and IME. This of course only works for applications that have a resizeable area that can be reduced to make enough space, but the vertical space in this mode is actually no less than what is available in landscape orientation, so very often an application can already accommodate it.

The final major mode is fullscreen or extract mode. This is used when the IME is too large to reasonably share space with the underlying application. With the standard IMEs, you will only encounter this situation when the screen is in a landscape orientation, although other IMEs are free to use it whenever they desire. In this case the application window is left as-is, and the IME simply displays fullscreen on top of it, as shown below.

Because the IME is covering the application, it has its own editing area, which shows the text actually contained in the application. There are also some limited opportunities the application has to customize parts of the IME (the "done" button at the top and enter key label at the bottom) to improve the user experience.

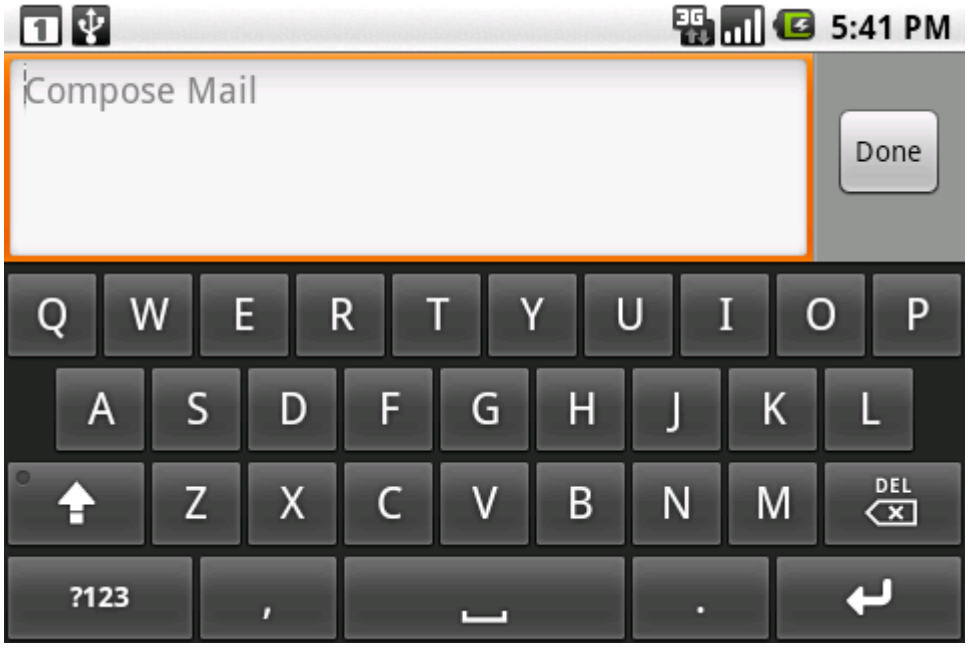


FIG. NO. 2.3 : FULL SIZED KEYBOARD

2.3 DRAWBACKS

While onscreen keyboards or soft key boards are suitable for latin based languages like English, due to its low number of characters, it is not very much suitable for Indic languages especially for Tamil. This is an example of a Tamil soft keyboard:

FIG. NO. 2.4 : TAMIL SOFT KEYBOARD

Clearly one could see the huge number of keys in keyboard, which is crammed up. Even though the keyboard provides support for all the Tamil Characters it does not give a good user experience and makes tough to type swiftly due to its very small key spacing. The keyboard does not have a consistent view in full screen editing mode and partial view editing mode.

2.4 PROPOSED SYSTEM

The proposed system aims to develop an engine which is capable of recognizing Tamil character handwritten strokes on the touch screen. The proposed system may not be accurate in recognition as that of the soft keyboard, due to computational constraints of the available Android devices. Therefore it is required to trade-off accuracy for saving time.

The issue of accuracy is handled rather well, with the provision of suggestion panel in the proposed system's UI, which displays the closest match of the stroke drawn by the user. In case if the proposed system commits a wrong input, it can be rectified by selecting one of the options available in the suggest panel.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

3.1.1 PURPOSE

"Bharathi" is a Handwriting Input Method Editor (IME) for Tamil script, targeted for Aakash. The IME recognizes characters through touch gestures. It will eliminate the difficulties faced by the native people in using keyboards.

3.1.2 SCOPE

This project is targeted on Android based Aakash tablets and concerns detecting free handwritten characters through touch gestures. This project is restricted to recognizing Tamil characters and numerals.

3.2 GENERAL DESCRIPTION

3.2.1 PRODUCT PERSPECTIVE

This product is an online Tamil handwritten character recognition **application for handheld devices which was done by a group of students** from the 15th batch of the Dept of Computer Science and Engineering, Saveetha Engineering College. However, this will be a self-contained product in itself. The product which was developed entirely for Aaakash-2 tablets.

3.2.2 PRODUCT FUNCTIONS OVERVIEW

PRE-PROCESSING:

The purpose of pre-processing is to discard irrelevant information in the input data, which can negatively affect the recognition. This concerns speed and accuracy. Pre-processing usually consists of normalization and smoothing.

FEATURE EXTRACTION

The neural network is presented a pattern, this could be an image and hand written data.

Feature extraction consists of three steps:

- Extreme coordinate measurement
- Grabbing character into grid
- Character digitization.

The handwritten character is captured by its extreme coordinates from left /right and top/bottom and is subdivided into a rectangular grid of specific rows and columns.

Then it searches the presence of character pixels in every box of the grid. The boxes found with character pixels are considered “on” (1) and the rest are marked as “off” (0). Give the user the ability to click a button and enable the product to start receiving inputs (action listener). Product should be compatible with the full Tamil alphabet.

CLASSIFICATION

Classification is done using the already trained SOM based engine. For proper classification of the strokes it is necessary that the neural net is trained properly properly preprocessed training set.

3.2.3 USER CHARACTERISTICS

- The user is native person who lacks knowledge of operating traditional keyboards.
- Intended user no need to have any background knowledge of the application.
- All users can use Bharathi in their own natural handwriting.
- Users can directly write on touch screens rather than using hard keyboards.

3.2.4 OPERATING ENVIRONMENT

HARDWARE SPECIFICATION

TABLE 3.1: HARDWARE SPECIFICATION

Target	Processor	RAM	Screen Size
Aakash tablet	Cortex A8 1.0 GHz	512 MB	7”

SOFTWARE SPECIFICATION

TABLE 3.2: SOFTWARE SPECIFICATION

FrontEnd	Tools Used	OS	VM
Android JAVA Processing	Android SDK	Android	Dalvik

3.2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

Tamil Language Related Constrains: Tamil Alphabet consists of 256 symbols, comprising of 18 vowels, 12 consonants and 1 semi consonants. Apart from these characters they can have several modifiers such as grantha script etc. Most of the characters are curved in nature.

Software constraints: It can be implemented only on Android Environments. It will be specially tailored to fit Aakash.

Hardware constraints: This require special Hardware as This product is only valid if a touch based tablet is available for the user minimum of 512MB RAM and a Hard disk space of 10MB .

3.2.6 ASSUMPTIONS AND DEPENDENCIES

- To obtain higher accuracy user must have standard style of writing tamil characters.

- User must possess, and be able to use a pressure sensitive touch pad.

3.3 FEASIBILITY ANALYSIS

3.3.1 ECONOMIC FEASIBILITY

In the system, the user is most satisfied by economic feasibility. Because, to implement this system, only expects the user to have an Android device with touch screen capability.

Once the system is ready for release it is anticipated that it would be released under end user friendly open source license(s), thereby making it free.

3.3.2 TECHNICAL FEASIBILITY

The system needs no substantial changes in the end user's device. It only requires one step install of the application in the user's device. The system may require 5 MB of SD card space for the sake of storing the Engine and for future updation of the application.

3.3.3 SOCIAL FEASIBILITY

Bharathi is believed to have high social acceptance, due to its social centric ideology. One has to note that Bharathi is enabling vernacular language support, which is very much needed for the native users. It is estimated that this vernacular support system will reach all sections of the society.

3.4 SYSTEM FEATURES

The product should support or encompass the following features,

- The functionalities should be pluggable / work in hand to hand with existing IME.
- The menu bar should include a small guide for the user and requires a drag bar to adjust the speed at which the character recognition is activated.

- Dynamically give the user a list of possible letters when the character is being drawn.
- Give the user a minimal list of probable letters if a character is not recognized

3.5 EXTERNAL INTERFACE REQUIREMENTS

3.5.1 USER INTERFACES

When using Bharathi, the users will use stylus/fingers to write the letters. The software interface will be in the form of a toolbar or a tool box.

3.5.2 HARDWARE INTERFACES

The main hardware component used in Bharathi is the Aakash 2 tablet.

3.5.3 SOFTWARE INTERFACES

Bharathi should be compatible with Android, and should be pluggable with other apps.

3.6 OTHER NONFUNCTIONAL REQUIREMENTS

3.6.1 PERFORMANCE REQUIREMENTS

The character recognition should be real time. While a user is writing a letter on the Stylus/fingers, Bharathi should be able to recognize possibilities of the character which is being written. This should happen in such a way that when the user finishes writing the character the software have to provide the character which was written without any delay. The software should not use 100% of the available CPU or RAM. This is because users might use some other applications are running. So the CPU and RAM usage will have to be at a level where several other applications can run without disruption.

3.6.2 SAFETY REQUIREMENTS

Bharathi currently does not access any system critical components, although due to its high compution requirement, it may overshoot the memory; therefore memory handling modules are needed to work around

the issue. The system should allow other input method editor to take over under such critical conditions.

It is important to note that Bharathi has the capability to access user's inputs which include passwords and highly sensitive data; care must be taken so that Bharathi never stores such data.

If a need arises to require any network access, Bharathi should notify the user's about the need for using the resource and wait for user's approval.

3.6.3 SOFTWARE QUALITY ATTRIBUTES

RELIABILITY

Bharathi should be able to identify Tamil handwritten characters with maximum accuracy possible without tampering the speed of recognition. At any stage speed should not be compensated for accuracy. An attempt should be made to recognize all characters.

PORTABILITY

The utility can be run on any Android device which has touch input capability, the system heavily depends on Input Method Framework, and therefore it is IMF is required to run Bharathi.

MAINTAINABILITY

The application does not require any special maintanence, an update system should be developed for the delivery of better recognition engines.

AVAILABILITY

The application can run 24×7 without any restriction.

CHAPTER 4

DETAILED DESIGN

4.1 INTRODUCTION

This chapter includes the architecture, use case, activity, collaboration, sequence, dataflow diagrams related to the system design.

4.2 LIFE CYCLE OF THE IME

FIG. NO. 4.1: THE LIFECYCLE DIAGRAM OF IME

Lifecycle diagram is the core concept behind the application, the application goes through the above steps each and every time the application is activated.

4.3 ARCHITECTURE OF BHARATHI

The application follows a simple architecture. Most of the architectural needs are already handled by the Android's Input method frame work and the Android' s system libraries. It is only reqired to extend the UI capabilities of the android's IMF and introduce computational capabilites by implementing SOM.

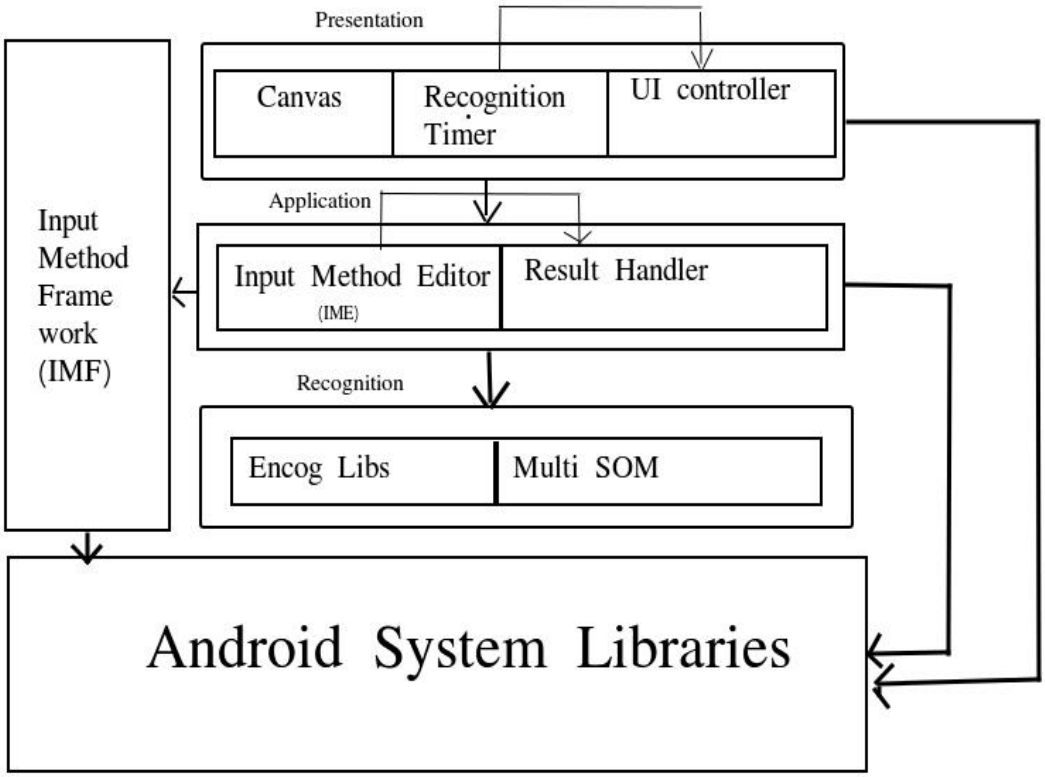


FIG. NO. 4.2 : ARCHITECTURE DIAGRAM OF BHARATHI

PRESENTATION LAYER

The presentation layer has three sub-components:

CANVAS

The canvas is the drawing area in which the user draws the input stroke. It controls the thickness, color, ant-alaising and other aesthetic properties of the drawing area.

RECOGNITION TIMER

It is responsible for time keeping. It has an internal timer which triggers the recognition action. The timer is adjustable and can be manipulated by the user via UI controller.

UI CONTROLLER

The UI controller provides facility for customization. It can enable users to switch between recognition engines and other customizations like adjusting the recognition timer.

APPLICATION LAYER

The application layer takes care of the coordination between the presentation layer and the Recognition layer:

IME

As discussed in the earlier sections the IME is responsible for the basic lifecycle of the application. It provides the basis for the event handling during different situation that the app faces.

RESULT HANDLER

This component is responsible for the handling the result from the recognition layer. The results received are generally needed to be split into two, one for committing and the other for suggest panel, and such intricacies are handled by this layer.

RECOGNITION LAYER

As its name implies recognition layer is for recognizing the input stroke.

ENCOG LIBRARIES

Then encog libraries are the core of Bharathi, which has neural network specific support, which includes support for calculation of weights and other matrix computation components.

MULTI-SOM

The Multi SOM layer is the extension of the traditional SOM available in which is capable of giving multiple results based on the closeness with input stroke.

4.4 DATAFLOW DESIGN

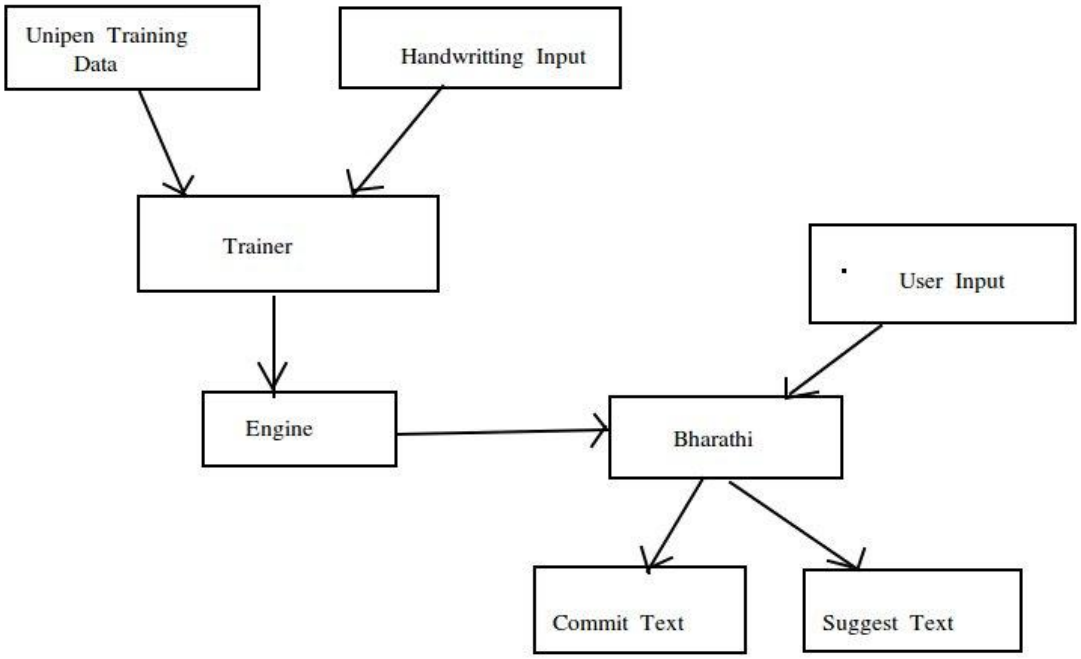


FIG. NO. 4.3 : DATA FLOW DIAGRAM FOR ENGINE

The data flow in the application is depicted by the above figure. The typical data flow begins with the unipen training set and custom handwriting input. These inputs are sent to the trainer for the generation the recognition engine.

With combined data from the engine and the user input strokes, the SOM algorithm in the Bharathi application outputs the commit text, which the application assumes that the recognition is correct and the suggest texts which are closest match to the commit text.

4.5 UML DIAGRAMS

4.5.1 USE CASE DIAGRAMS

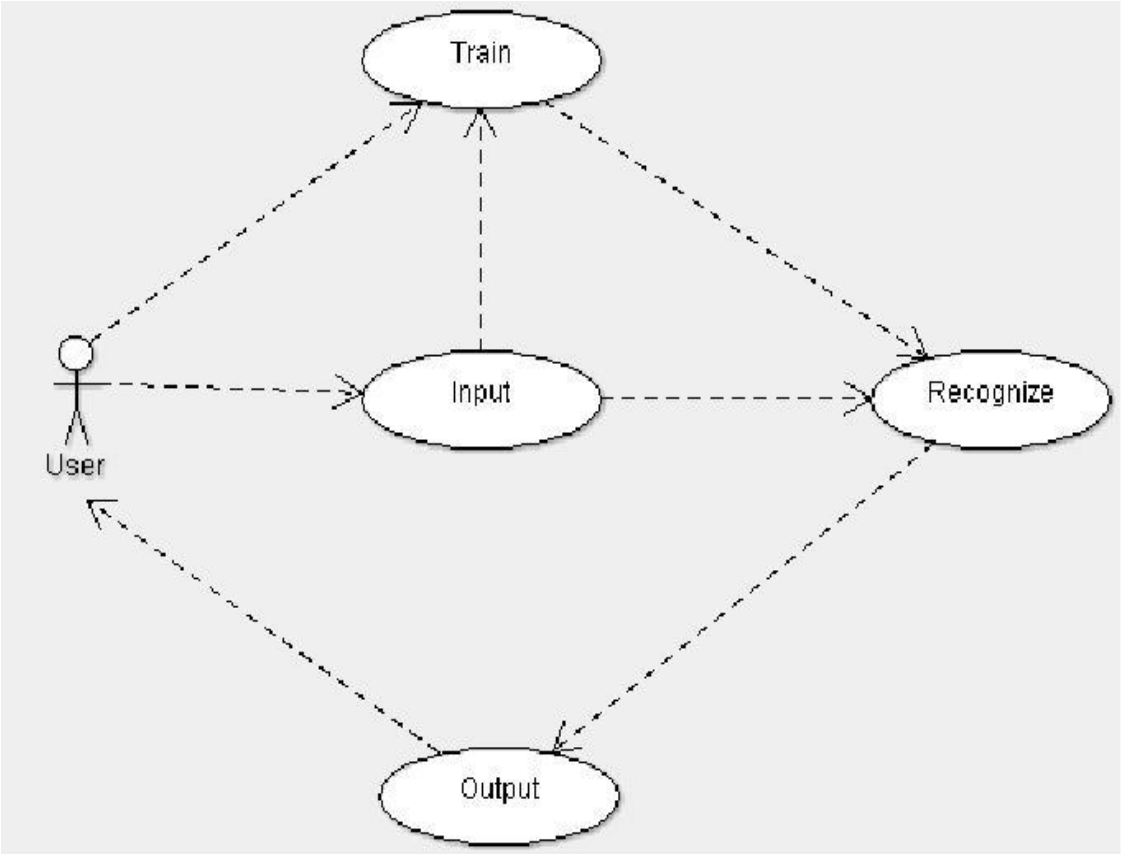


FIG. NO. 4.4: USE CASE DIAGRAM FOR IME

4.5.2 CLASS DIAGRAM

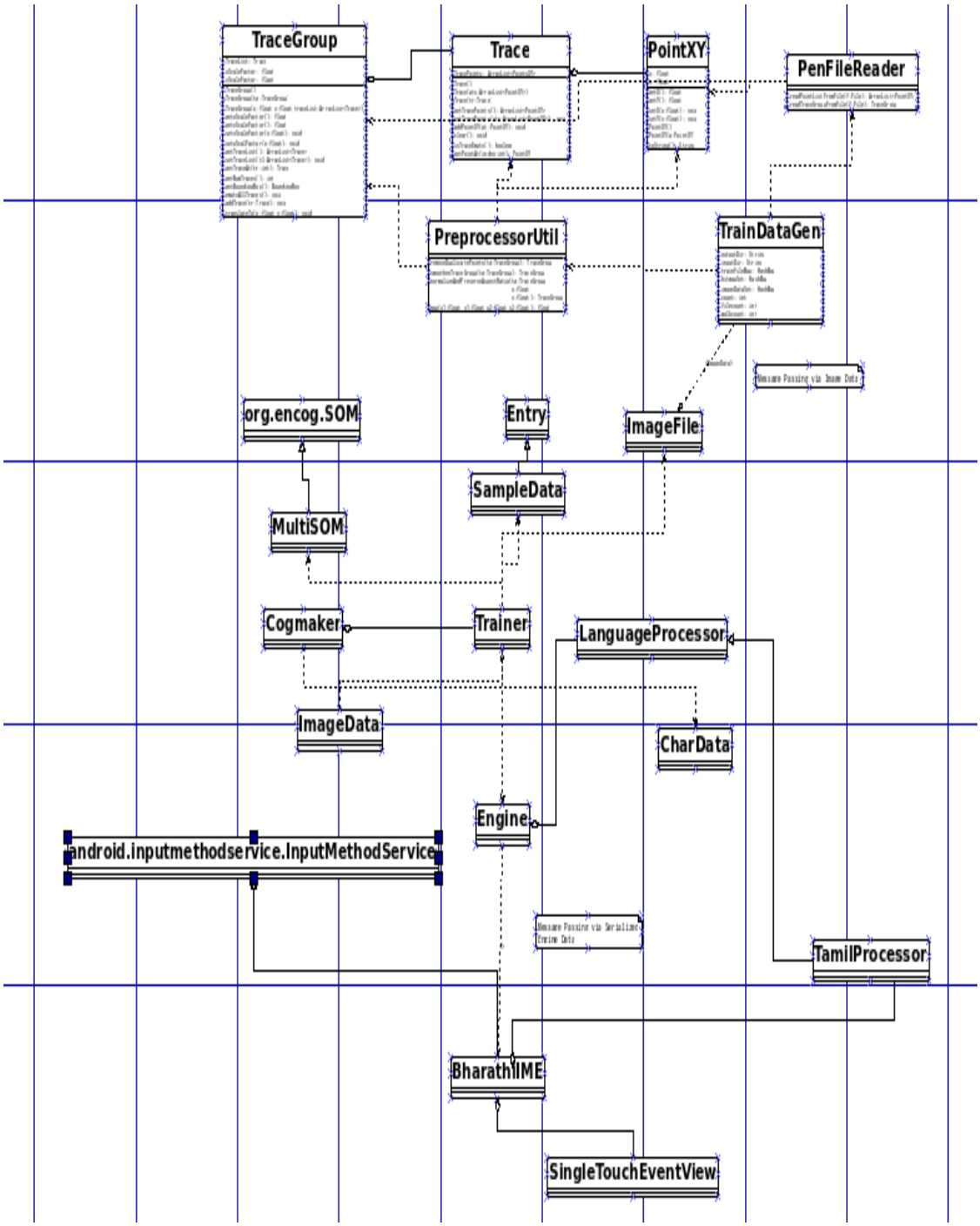


FIG. NO. 4.5 : CLASS DIAGRAM FOR BHARATHI

4.5.3 ACTIVITY DIAGRAMS

RECOGNIZE ACTIVITY

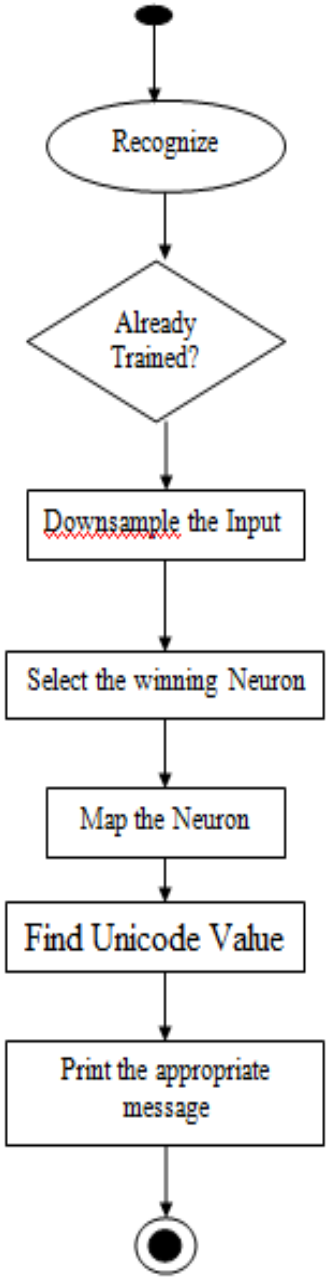


FIG. NO. 4.6 : RECOGNIZE ACTIVITY DIAGRAM FOR ENGINE

TRAINING ACTIVITY

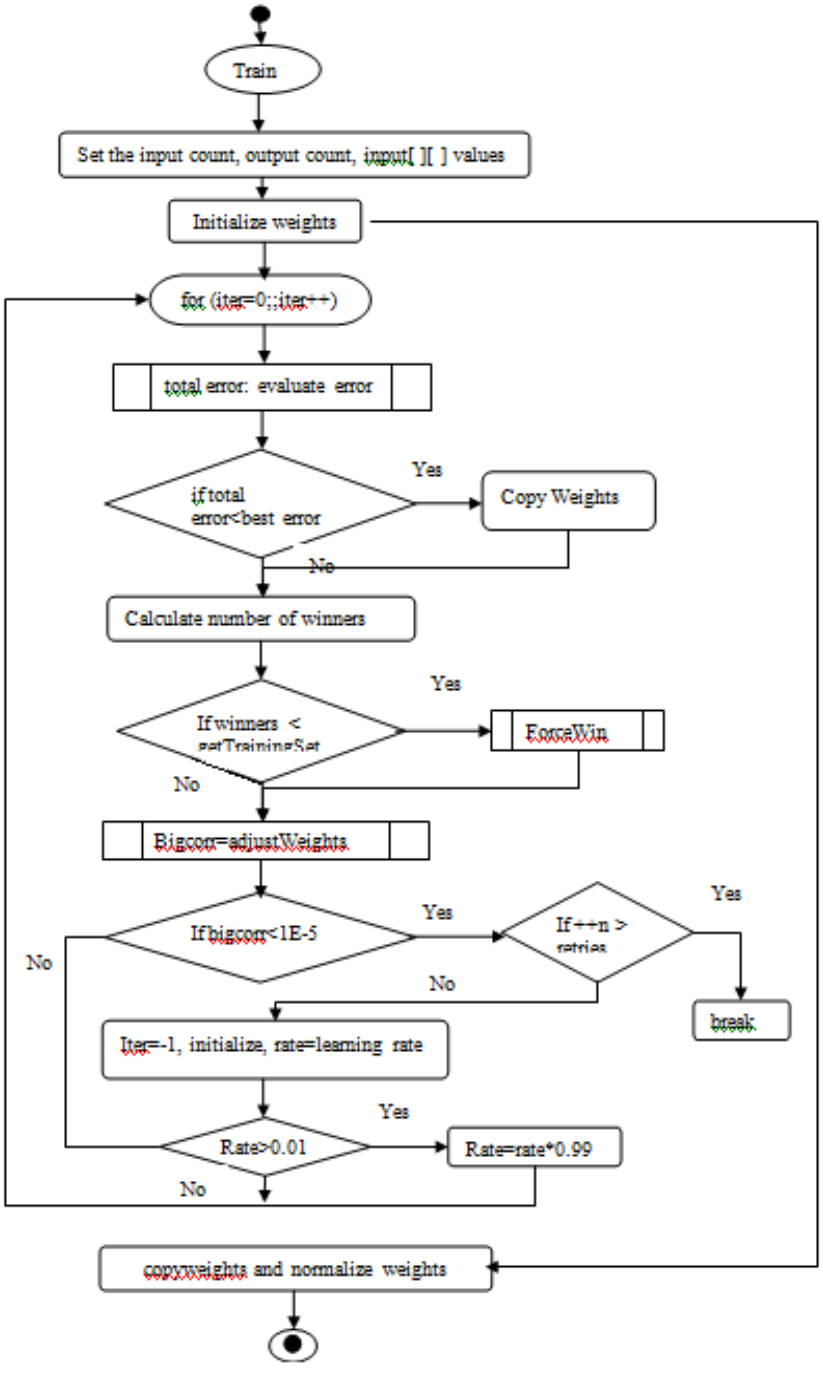


FIG. NO. 4.7 : TRAINING ACTIVITY DIAGRAM FOR ENGINE

4.5.4 SEQUENCE DIAGRAMS

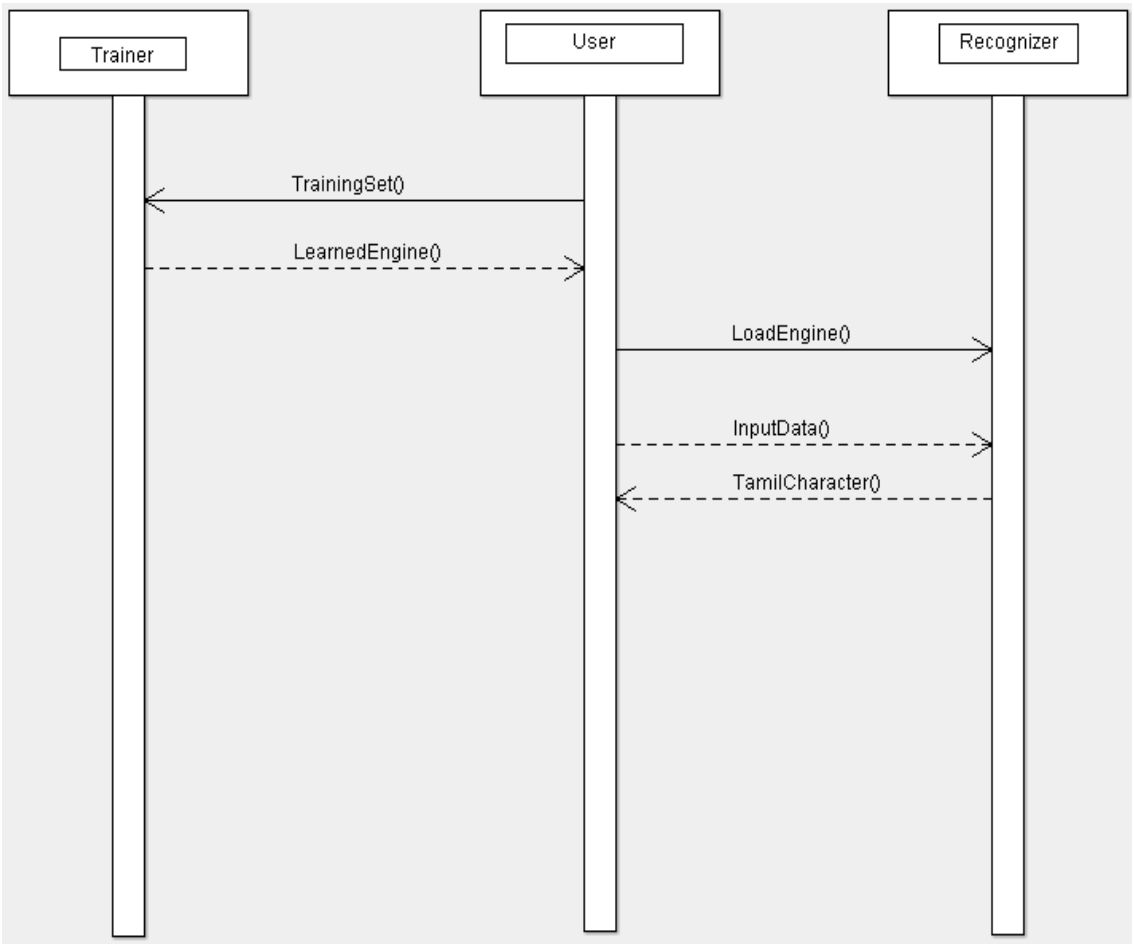


FIG. NO. 4.8 : SEQUENCE DIAGRAM FOR BHARATHI

4.4.5 COMPONENT DIAGRAM

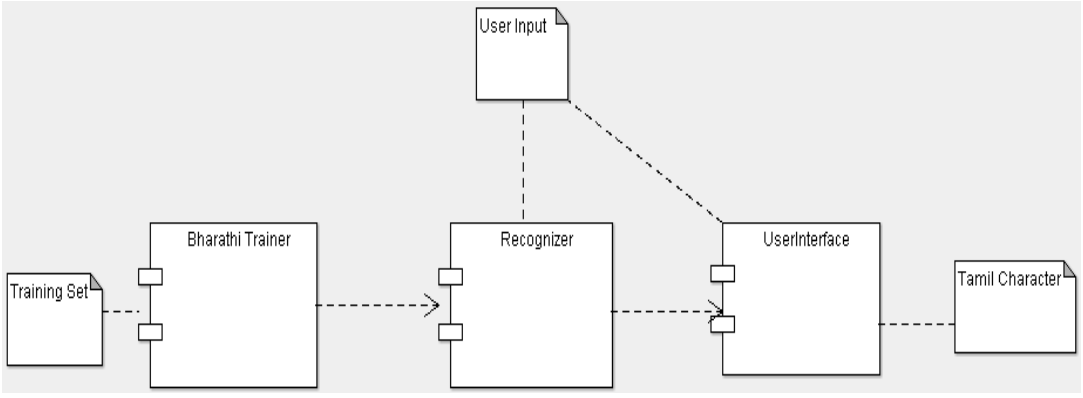


FIG. NO. 4.9 : COMPONENT DIAGRAM FOR BHARATHI

CHAPTER 5

MODULES

5.1 TRAINING MODULE

```
private void trainSOM() {
final int inputNeuron = Trainer.DOWNSAMPLE_HEIGHT
                        * Trainer.DOWNSAMPLE_WIDTH;
final int outputNeuron = this.letterList.size();
final MLDataSet trainingSet = new BasicMLDataSet();
for (int t = 0; t < this.letterList.size(); t++) {
final MLData item = new BasicMLData(inputNeuron);
int idx = 0;
final SampleData ds = this.letterList.get(t);
for (int y = 0; y < ds.getHeight(); y++) {
for (int x = 0; x < ds.getWidth(); x++) {
    item.setData(idx++, ds.getData(x, y) ? .5 : -.5);
                                        }
                                        }
trainingSet.add(new BasicMLDataPair(item, null));
                                        }
this.net = new MultiSOM(inputNeuron,outputNeuron);
this.net.reset();
SOMClusterCopyTraining train = new SOMClusterCopyTraining
(this.net, trainingSet);
train.iteration();
```

```
        showMessageBox("Training has completed." +
trainingSet.getRecordCount());

        System.out.println("Training has completed.");

        //neuronMap = mapNeurons();

        neuronMap = makeNeuronMap();

}
```

This module takes all the characters in the database as the input. Firstly, the input neuron and output neuron count and also the input values for each character is set, then the weights are initialized using. Next, the neurons are trained in an infinite loop to get the winning neuron of each character in the database. The loop runs until the error rate is acceptable and the desired correction is obtained.

5.2 RECOGNIZE MODULE

```
public CharData[] recognizeAction(int count) {
                if (this.net == null) {
                        //Log.e(TAG, "I need to be trained first!");
                        return null;
                }
                if(count>this.engine.getNeuronMap().length) {
                        count = this.engine.getNeuronMap().length;
                }
                if(entry==null) {
                        init();
                }
                this.entry.downsample(getImagePixels());
```

```
        final MLData input = new
BasicMLData(Ocr.DOWNSAMPLE_HEIGHT*
Ocr.DOWNSAMPLE_WIDTH);

        int idx = 0;

        final SampleData ds = this.entry.getSampleData();

        for (int y = 0; y < ds.getHeight(); y++) {

                for (int x = 0; x < ds.getWidth(); x++) {

                        input.setData(idx++, ds.getData(x, y) ? .5 : -.5);
                }

        }

        final int[] result = this.net.matches(input,
count);

        List<CharData> retList = new    ArrayList<CharData>();

        for(int i=0; i<count; i++) {

                CharData newData = neuronMap[result[i]];

                if(!retList.contains(newData)) {

                retList.add(newData);

                }

        }

        CharData[] ret = retList.toArray(new CharData[]
{});           clearAction();


        return ret;

}
```

This module takes the character drawn on the drawing panel as the input. It firstly checks whether the character drawn has been previously trained or not. Then the character drawn is downsampled and the input values are set which are given to the kohonen network to get the winning neuron. Later the input values are set for every character in the database which are given to the kohonen network to get the winning neuron of each character. Next the index of the winning neuron of the character drawn is mapped with the index of the winning neuron of each character and checks for the unicode range to print the appropriate message (Ex:-“Tamil letter : entered character”).

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

Bharathi is currently capable of recognizing 44 Tamil characters, one has to note that the project is not release ready. The project needs to be trained with more characters, so that the expected reliability is achieved.

6.2 FUTURE ENHANCEMENTS

As a future enhancement, we plan to optimize the engine for increased efficiency and performance. To take the project to next level, the Bharathi should support word prediction. We also plan to extend this project to support several Indic languages.

APPENDIX A

DEVELOPMENTAL TOOLS

A.1 ANDROID SDK

The Android Developer Tools (ADT) plugin for Eclipse provides a **professional-grade development environment for building Android apps.** It's a full Java IDE with advanced features to help you build, test, debug, and package your Android apps. Moreover its free, open-source, and runs on most major OS platforms. A copy of this tool can be obtained from, 'http://developer.android.com/sdk/ index.html'.

A.2 JAVA

Java is a set of several computer software products and specifications from Sun Microsystems (which has since merged with Oracle Corporation), that together provide a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common, Java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web on desktop computers.

A.3 PROCESSING

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production. User can download Processing from 'http://processing.org/download/'.

A.4 ENCOG

Encog is an advanced machine learning framework that supports a variety of advanced algorithms, as well as support classes to normalize and

process data. Machine learning algorithms such as Support Vector Machines, Artificial Neural Networks, Genetic Programming, Bayesian Networks, Hidden Markov Models and Genetic Algorithms are supported. Most Encog training algoritms are multi-threaded and scale well to multicore hardware. Encog can also make use of a GPU to further speed processing time. A GUI based workbench is also provided to help model and train machine learning algorithms.

APPENDIX B

RUNTIME ENVIRONMENT

B.1 ANDROID

Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first Android-powered phone was sold in October 2008.

Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language. In October 2012, there were approximately 700,000 apps available for Android, and the estimated number of applications downloaded from Google Play, Android's primary app store, was 25 billion.

These factors have allowed Android to become the world's most widely used smartphone platform, overtaking Symbian in the fourth quarter of 2010, and the software of choice for technology companies who require a low-cost, customizable, lightweight operating system for high tech devices without developing one from scratch. As a result, despite being primarily designed for phones and tablets, it has seen additional applications on televisions, games consoles, digital cameras and other electronics. Android's open nature has further encouraged a large community of developers and enthusiasts to use the open source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially released running other operating systems.

Android had a worldwide smartphone market share of 75% during the third quarter of 2012, with 750 million devices activated in total and 1.5 million activations per day.

B.2 ANDROID TECHNICAL INFORMATION

DALVIK TECHNICAL INFORMATION

The Dalvik Virtual Machine is the heart of Android. It's a fast, just-in-time compiled, optimized bytecode virtual machine. Android applications are compiled to Dalvik bytecode and run on the Dalvik VM.

DEBUGGING

Android is a large and complex system.

ENCRYPTION TECHNICAL INFORMATION

The Android Open-Source Project includes the ability to encrypt the user's data. It covers the few things that must be done so encryption will work.

SECURITY TECHNICAL INFORMATION

Android provides a robust multi-layered security architecture that provides the flexibility required for an open platform, while providing protection for all users of the platform.

INPUT TECHNICAL INFORMATION

Android's input subsystem is responsible for supporting touch screens, keyboard, joysticks, mice and other devices.

DATA USAGE TECHNICAL INFORMATION

Android's data usage features allow users to understand and control how their device uses network data.

ACCESSORY PROTOCOL INFORMATION

Android devices can connect to hardware accessories, such as audio docks, keyboards and custom hardware, through USB or Bluetooth.

EXTERNAL STORAGE TECHNICAL INFORMATION

Android supports devices with external storage, typically provided by physical media or an emulation layer.

APPENDIX C

SAMPLE CODE

C.1 BharathiIME.java

```
public class BharathiIME extends InputMethodService{

//declarations

@Override

public void onCreate() {

// TODO Auto-generated method stub

super.onCreate();

}

@Override

public void onInitializeInterface() {

// TODO Auto-generated method stub

super.onInitializeInterface();

}

private void showMenu() {

menuView.setVisibility(View.VISIBLE);

menuVisible = true;

}

private void hideMenu() {
```

```
menuView.setVisibility(View.GONE);

menuVisible = false;

}

private void populateMenu() {

TextView[] items = new TextView[] {

makeMenuItem("Help", null),

makeMenuItem("Settings", null),

makeMenuItem("Tamil",
"com.jagmakara.bharathi.lang.TamilProcessor"),

makeMenuItem("Numeric",
"com.jagmakara.bharathi.lang.NumberProcessor"),

};

for(TextView item : items) {

menuView.addView(item);

}

}

private TextView makeMenuItem(final String menuText, final
String classname) {

final SharedPreferences prefs =
this.getSharedPreferences(stupidpenPrefs,
Context.MODE_PRIVATE);

TextView menuItem = new TextView(this);
```

```
menuItem.setMinimumHeight(40);

menuItem.setMinimumWidth(70);

menuItem.setTextAppearance(this, R.style.menuText);

menuItem.setCompoundDrawablesWithIntrinsicBounds(0, 0, 0,
R.drawable.horizontal_line);

menuItem.setText(menuText);

menuItem.setGravity(Gravity.LEFT&Gravity.CENTER_VERTIC
AL);

menuItem.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

if(menuText.equalsIgnoreCase("help")) {

//show help

Intent intent = new Intent();

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

intent.setClassName(PACKAGE_NAME, HELP_ACTIVITY);

startActivity(intent);

} else if(menuText.equalsIgnoreCase("settings")) {

//call activity using intent

Intent intent = new Intent();

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
intent.setClassName(PACKAGE_NAME,
SETTINGS_ACTIVITY);
startActivity(intent);
} else {
try {
loadProcessor(classname);
prefs.edit().putString(defaultKeyboard, classname).commit();
} catch (Exception e) {
//Log.e(TAG, e.getMessage(), e);
}
}
hideMenu();
}
});
return menuItem;
}
private void dotButtonAction() {
CharData dot = new CharData(".");
dot.setAlign(0);
ocr.dumbProcess(dot);
```

```
}
private void spaceButtonAction() {
CharData dot = new CharData(" ");
dot.setAlign(0);
ocr.dumbProcess(dot);
}
private void enterButtonAction() {
CharData dot = new CharData("\n");
dot.setAlign(0);
ocr.dumbProcess(dot);
}
private void backspButtonAction() {
CharData dot = new CharData("");
dot.setAlign(0);
ocr.dumbProcess(dot);
getCurrentInputConnection().deleteSurroundingText(1, 0);
ocr.showSliceText();
}
@Override
public View onCreateInputView() {
```

```
rootView = getLayoutInflater().inflate(R.layout.ime, null);

ocr = (Ocr) rootView.findViewById(R.id.writePad);

stackView = (TextView) rootView.findViewById(R.id.stackView);

suggestionsViewGroup = (LinearLayout) rootView.findViewById(R.id.suggestionsViewGroup);

sliceView = (TextView) rootView.findViewById(R.id.sliceView);

movableView = (LinearLayout) rootView.findViewById(R.id.movableView);

ocr.setImeService(this);

View showMenuButton = rootView.findViewById(R.id.showMenuButton);

menuView = (LinearLayout) rootView.findViewById(R.id.menuView);

View dotButton = rootView.findViewById(R.id.dotButton);

View spaceButton = rootView.findViewById(R.id.spaceButton);

View enterButton = rootView.findViewById(R.id.enterButton);

View backspButton = rootView.findViewById(R.id.backspButton);

final View menuControlView = rootView.findViewById(R.id.controlsView);

this.populateMenu();

this.hideMenu();

dotButton.setOnTouchListener(new View.OnTouchListener() {
```

```
private Handler mHandler;
 @Override public boolean onTouch(View v, MotionEvent event)
{
 switch(event.getAction()) {
 case MotionEvent.ACTION_DOWN:
 if (mHandler != null) return true;
 mHandler = new Handler();
 mHandler.postDelayed(mAction, 500);
 break;
 case MotionEvent.ACTION_UP:
 if (mHandler == null) return true;
 mHandler.removeCallbacks(mAction);
  dotButtonAction();
  mHandler = null;
   break;
  }
  return true;
   }
    Runnable mAction = new Runnable() {
   @Override public void run() {
```

```
dotButtonAction();
mHandler.postDelayed(this, 200);
} };
});
spaceButton.setOnTouchListener(new View.OnTouchListener() {
private Handler mHandler;
 @Override public boolean onTouch(View v, MotionEvent event)
                                switch(event.getAction()) {
                                case MotionEvent.ACTION_DOWN:
                                   if (mHandler != null) return true;
                                   mHandler = new Handler();
                                   mHandler.postDelayed(mAction, 500);
                                   break;
                                case MotionEvent.ACTION_UP:
                                   if (mHandler == null) return true;
                                   mHandler.removeCallbacks(mAction);
                                   spaceButtonAction();
                                   mHandler = null;
                                   break;
                                }
```

```
                return true;
            }
            Runnable mAction = new Runnable() {
                @Override public void run() {
spaceButtonAction();
                    mHandler.postDelayed(this, 200);
                }
            };
        });
enterButton.setOnTouchListener(new View.OnTouchListener() {
private Handler mHandler;
@Override public boolean onTouch(View v, MotionEvent event) {
                switch(event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    if (mHandler != null) return true;
                    mHandler = new Handler();
                    mHandler.postDelayed(mAction, 500);
                    break;
                case MotionEvent.ACTION_UP:
                    if (mHandler == null) return true;
```

```
                    mHandler.removeCallbacks(mAction);
                    enterButtonAction();
                    mHandler = null;
                    break;
                }
                return true;
            }
            Runnable mAction = new Runnable() {
                @Override public void run() {
                enterButtonAction();
                    mHandler.postDelayed(this, 200);
                }
            };
        });
backspButton.setOnTouchListener(new View.OnTouchListener() {
private Handler mHandler;
@Override public boolean onTouch(View v, MotionEvent event) {
                    switch(event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                    if (mHandler != null) return true;
```

```
                    mHandler = new Handler();
                    mHandler.postDelayed(mAction, 500);
                    break;
                case MotionEvent.ACTION_UP:
                    if (mHandler == null) return true;
                    mHandler.removeCallbacks(mAction);
                    backspButtonAction();
                    mHandler = null;
                    break;
                }
                return true;
            }
            Runnable mAction = new Runnable() {
                @Override public void run() {
                backspButtonAction();
                    mHandler.postDelayed(this, 100);
                }
            };
        });
    }
```

APPENDIX D

SAMPLE OUTPUT SCREENSHOTS

D.1 SETUP

MENU>>SETTINGS>>LANGUAGE AND INPUT>>BHARATIH

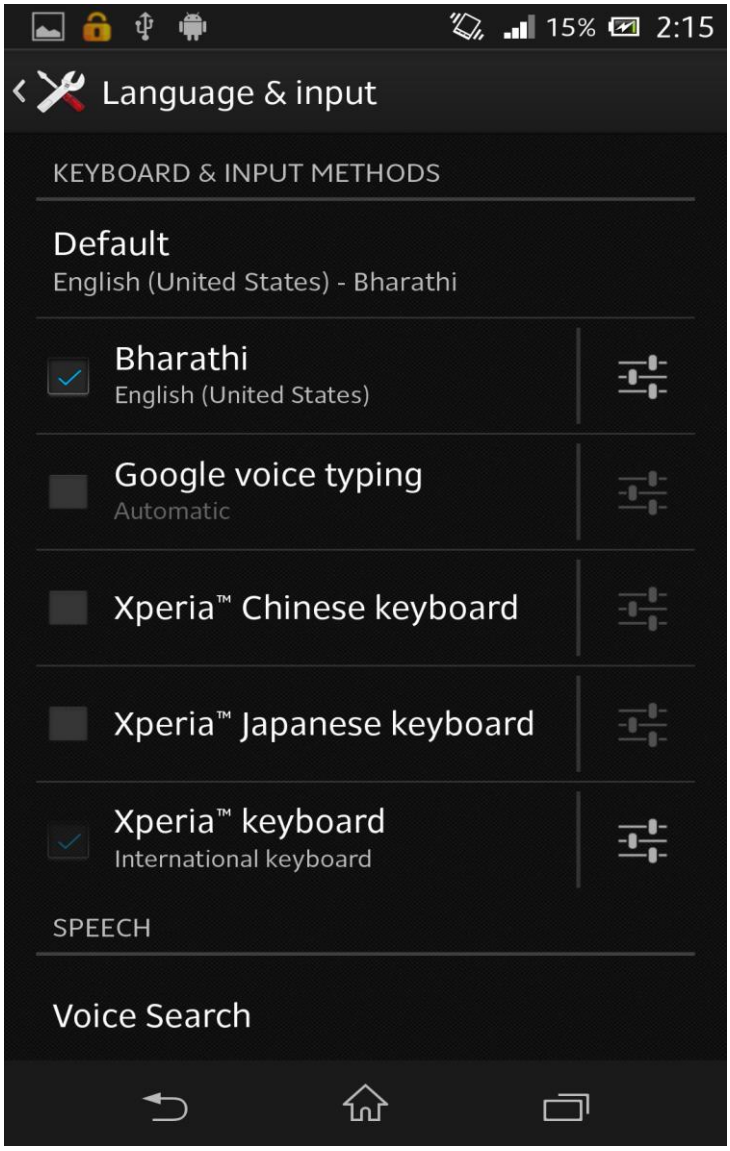
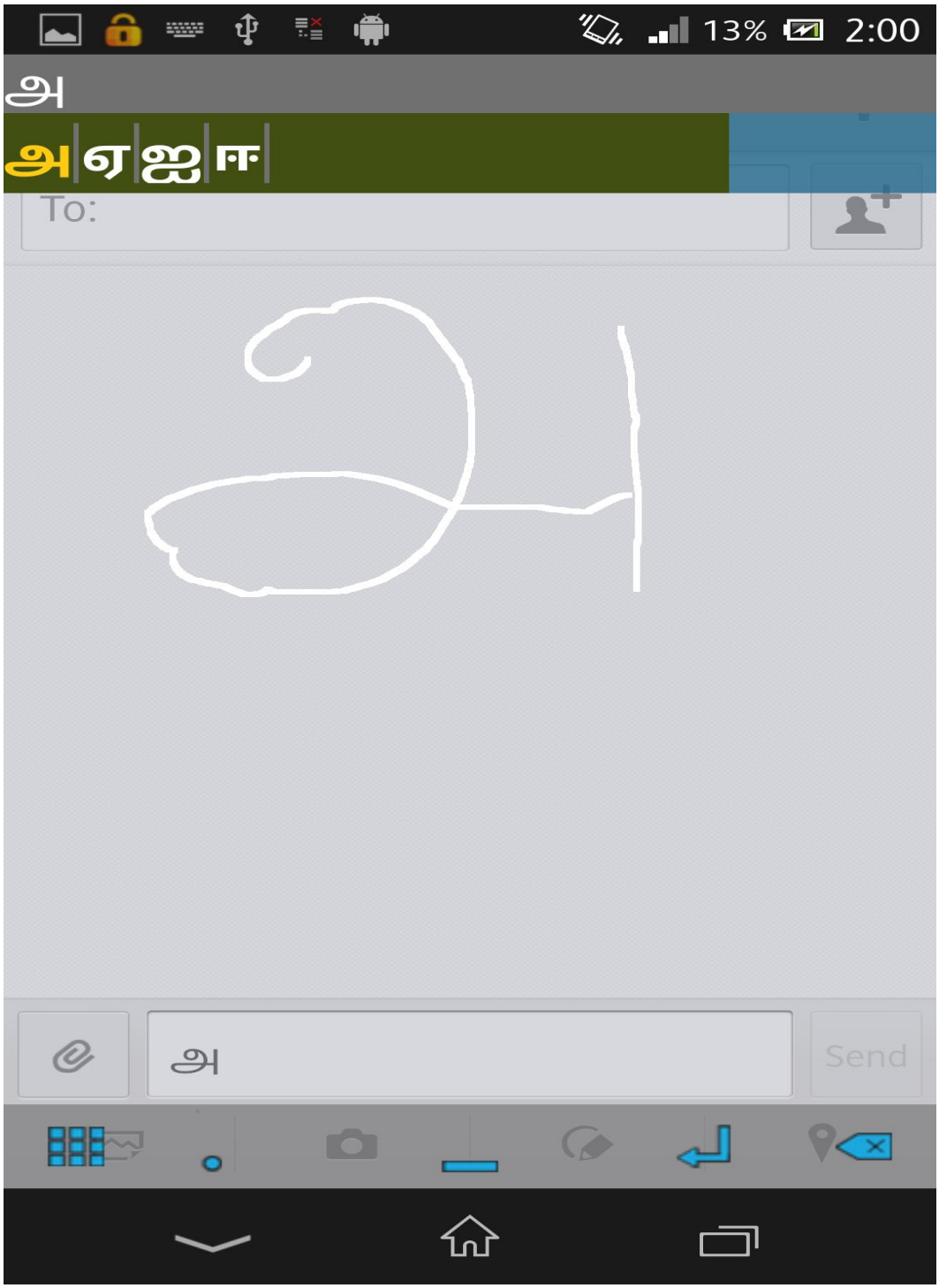


FIG. NO. D.1: SETTING UP BHARATHI

D.2 RECOGNIZE



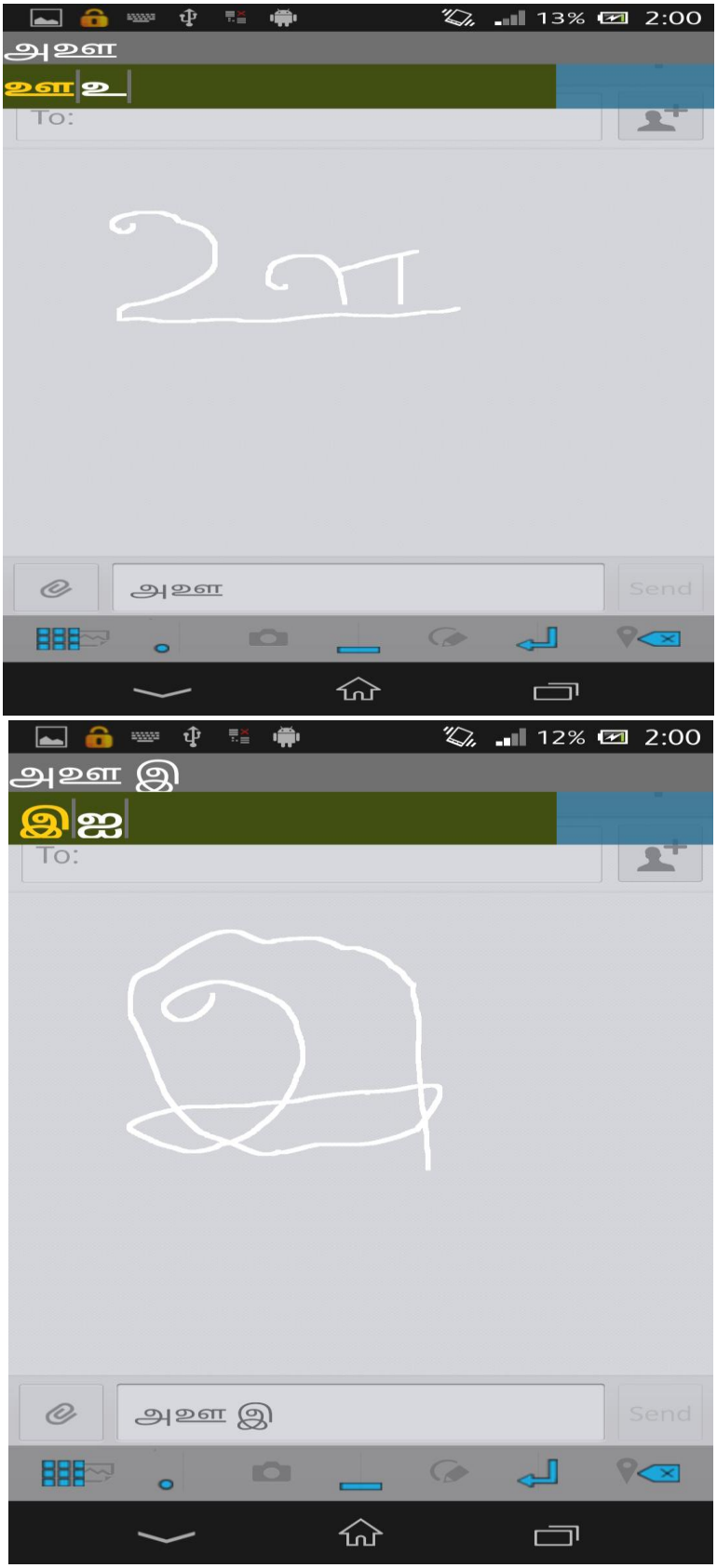


FIG. NO. D.2: IME SREENSHOTS

REFERNCES

- Android developer forum:

  'http://developer.android.com/sdk/index.html' and 'http://developer.android.com/guide/topics/text/creating-input-method.html' as on 28.04.2013.

- Craig, L. , (2004) 'Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition'.

- Encog Machine Learning Framework, 'https://github.com/encog'.

- Processing Language, 'http://processing.org/learning/'.

- Tamil Script, 'http://en.wikipedia.org/wiki/Tamil_script'.

- The Unicode Consortium: The Unicode Standard Version 5.0. Addison Professional, MA (2006).