

EFFECTIVE EMOTION TRANSPLANTATION IN AN END-TO-END TEXT-TO-SPEECH SYSTEM

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Technology degree in Information Technology

by

J. JAYA RASIKA (Reg. No. 37120034)

V.KAVIYA (Reg.No. 37120039)



**DEPARTMENT OF INFORMATION TECHNOLOGY SCHOOL OF
COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119.**

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **J. JAYA RASIKA (37120034),**
V. KAVIYA (37120039) who carried out the project entitled **"EFFECTIVE EMOTION**
TRANSPLANTATION IN AN END-TO-END TEXT-TO-SPEECH SYSTEM" under my supervision
from **August 2020** to **March 2021**.

Internal Guide

Dr.P.JEYANTHI, M.E.,Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **J.JAYA RASIKA(37120034)**, **V.KAVIYA(37120039)** hereby declare that the Project Report entitled **“EFFECTIVE EMOTION TRANSPLANTATION IN AN END-TO-END TEXT-TO-SPEECH SYSTEM”** is done by me under the guidance of **Dr.P. JEYANTHI, M.E., Ph.D**, Department of Information Technology at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Information Technology.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr.R.Subhashini, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P. JEYANTHI, M.E., PhD.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

EFFECTIVE EMOTION TRANSPLANTATION IN AN END-TO-END TEXT-TO-SPEECH SYSTEM

ABSTRACT:

In this paper, we propose an effective technique to transplant a source speaker's emotional expression to a new target speaker's voice within an end-to-end text-to-speech (TTS) framework. We modify an expressive TTS model trained using a source speaker's emotional speech database to reflect the voice characteristics of a target speaker for which only a neutral speech database is available. We set two adaptation criteria to achieve this. One criterion is to minimize the reconstruction loss between the target speaker's recorded and synthesized speech, such that the synthesized speech has the target speaker's voice characteristics. The other criterion is to minimize the emotion loss between the emotion embedding vectors extracted from the reference expressive speech and the target speaker's synthesized expressive speech. The proposed adaptation technique demonstrates more effective performance compared to conventional approaches in both quantitative and qualitative evaluations.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABTRACT	5
1	INTRODUCTION	9
	1.1 OUTLINE OF PROJECT	9
2	LITERATURE SURVEY	10
	2.1 LITERATURE SURVEY IN TABLE	10
3	METHODOLOGY	11
	3.1 SPEECH RECOGNITION	11
	3.2 EXISTING SYSTEM	11
	3.3 DISADVANTAGES	12
	3.4 PROPOSED SYSTEM	12
	3.5 ADVANTAGES	12
	3.6 SPEECH SYNTHESIS	13
	3.6.1 BLOCK DIAGRAM	13
	3.7 MODULE DESCRIPTION	13
	3.7.1 SPLASH ACTIVITY	14
	3.7.2 MAIN ACTIVITY	14
	3.7.3 SPEECH ACTIVITY	15
4	SOFTWARE DEVELOPMENT METHODOLOGY	16
	4.1 SYSTEM REQUIREMENTS	16
	4.1.1 SOFTWARE SPECIFICATION	16
	4.1.2 HARDWARE SPECIFICATION	16
	4.2 ANDROID	16
	4.3 FEATURES	17
	4.4 STORAGE	17

	4.5 ANDROID RUNTIME	18
	4.6 ARCHITECTURE	20
	4.6.1 ANATOMY OF AN ANDROID APPLICATION	20
	4.7 ANDROID BUILDING BLOCKS	21
	4.8 LIFECYCLE OF ACTIVITY	22
	4.9 OVERVIEW OF XML	23
	4.9.1 ADVANTAGES	23
	4.10 XML INTRODUCTION	23
	4.10.1 XML PARSERS	23
	4.10.2 XML PARSING PROCESS	23
	4.11 CREATE ANDROID APPLICATION	24
	4.12 ANDROID STUDIO	25
	4.12.1 TOOL WINDOWS	25
	4.13 JAVA PRPGRAMMING LANGUAGE	26
	4.13.1 HISTORY	26
	4.13.2 PRIMARY GOALS	27
	4.14 THE JAVA PROGRAMMING LANGUAGE	27
	4.14.1 THE JAVA PLATFORM	28
	4.14.2 JAVA RUNTIME ENVIRONMENT	30
	4.14.3 USES OF JAVA	30
5	RESULTS AND DISCUSSION	31
6	CONCLUSION AND FUTURE WORK	36
	REFERENCE	37
	APPENDIX	38
	A. PALGIARISM REPORT	38
	B. JOURNAL PAPER	39
	C. SOURCE CODE	42

FIG NO	FIGURE NAME	PAGE NO
2.1	BLOCK DIAGRAM	13
4.5	Architecture	20
4.8	Lifecycle of Activity	22
4.11	Create Android Application	24
4.14	An overview of the software development process	28
4.14	Through the java VM, the same application capable of running on multiple platform	28
4.14.1	The API and java virtual machine inaulate the program from the underlying hardware	29
5.1	Splash Screen	31
5.2	Main Screen	32
5.3	Strat Speech Screen	33
5.4	Fear Screen	34
5.5	Sadness Screen	35

Chapter 1

INTRODUCTION

1.1 OUTLINE OF PROJECT

This dissertation describes the creation of a framework for performing real-time analysis of speech for detecting emotional states and assessing the quality of speech. It shows how the classification is accomplished in real-time and how the system outperforms recent research in speech emotion detection. Both the core and the extension part of the project were completed on schedule.

Speech is one of the basic and natural way of communication among human beings. Emotions makes speech more expressive and effective. Different ways like laughing, yelling, teasing and crying. In order to recognition emotions, one first needs a precise idea of what they are, that is a theory of emotions. Emotion theories have a long tradition in psychology, having produced elaborate models that can be used as basis for automatic emotion recognition.

The most relevant ones in view of speech are presented in the first section of this chapter. They are also discussed regarding to what extent they are feasible to realize in practical applications, which is not fully the case for the more recent and complex theories.

Chapter 2

Literature Survey

2.1 Literature Survey In Table:

Paper Title	Technique Used	Merits	DE-Merits
Statistical parametric speech synthesis using deep neural networks	Statistical parametric speech synthesis, Hidden Markov model; Deep neural network	both systems could predict reasonable speech parameter trajectories	Minimum values in training data
Effective Spectral and Excitation Modeling Techniques for LSTM - RNN - Based Speech Synthesis Systems	Speech synthesis, improved time-frequency.	we analyze the train ability of the excitation parameters	However, it is limited in representing sequential characteristics
Efficient deep neural networks for speech synthesis using bottleneck features	Speaker adaptation techniques especially in HMM-based speech synthesis	Experimental results on the trainability, objective metrics and subjective listening	Average voice model using data from several speakers

Chapter 3

METHODOLOGY

3.1 SPEECH RECOGNITION:

Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a capability which enables a program to process human speech into a written format. While it's commonly confused with voice recognition, speech recognition focuses on the translation of speech from a verbal format to a text one whereas voice recognition just seeks to identify an individual user's voice.

Speech recognition incorporates different fields of research in computer science, linguistics and computer engineering. Many modern devices or text-focused programs may have speech recognition functions in them to allow for easier or hands-free use of a device.

3.2 EXISTING SYSTEM:

The task of generating natural speech from the input text, i.e., text-to-speech (TTS), is becoming increasingly important, as it is a key module in building human-computer interaction systems.

In addition, it is difficult to utter expressive speech while maintaining consistent expressiveness. For the same reason, the emotion loss function compares the expressiveness condition vectors instead of the target speaker's expressive speech data.

Even when there is only a small amount of expressive speech data of the target speaker, a target speaker's expressive TTS model can be obtained fairly easily by adapting the pre-trained model to minimize reconstruction loss. which is an error between recorded and synthesized expressive-speech of the target speaker.

3.3 DISADVANTAGES:

- ❖ Error speech translate
- ❖ Database issues
- ❖ Not qualitative evaluations.

3.4 PROPOSED SYSTEM:

The proposed adaptation technique demonstrates more effective performance compared to conventional approaches in both quantitative and qualitative evaluations.

TTS model can be obtained fairly easily by adapting the pre-trained model to minimize reconstruction loss, which is an error between recorded and synthesized

Expressive-speech of the target speaker. In this paper, we deal with the case in which the target speaker has only neutral speech data. In such applications, high-quality expressive speech databases with multiple voice identities are required. To deal with this, we effective emotion transplantation technique that guides the pre-trained model to preserve expressiveness characteristics during the adaptation process.

3.5 ADVANTAGES:

- ❖ Improved qualities' speech
- ❖ Not issues with database
- ❖ Finding easily identify speech

3.6 SPEECH SYNTHESIS:

- ❖ Speech synthesis is the artificial production of human speech
- ❖ A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software
- ❖ Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database

3.6.1 BLOCK DIAGRAM:

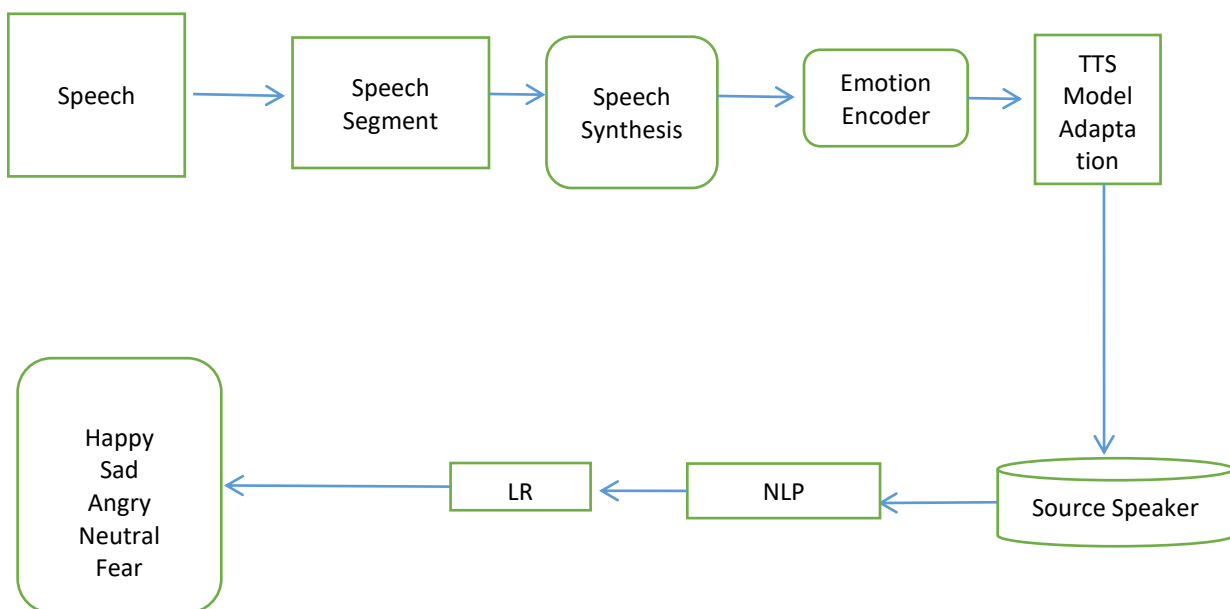


FIG: 3.6.1 BLOCK DIAGRAM

3.7 MODULE DESCRIPTION:

In our project we have only two modules. Those module description and modules are given below:

- ❖ Splash Activity
- ❖ Main Activity

❖ Speech Activity

3.7.1 Splash Activity:

Android Splash Screen is the first screen visible to the user when the application's launched. Splash screen is one of the most vital screens in the application since it's the user's first experience with the application.

Splash screens are used to display some animations (typically of the application logo) and illustrations while some data for the next screens are fetched.

The purpose of a Splash Screen is to quickly display a beautiful screen while the application fetches the relevant content if any (from network calls/database). With the above approach, there's an additional overhead that the Splash Activity uses to create its layout.

It'll give rise to slow starts to the application which is bad for the user experience (wherein a blank black/white screen appears).

3.7.2 Main Activity:

The Activity class is designed to facilitate this paradigm. When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. In this way, the activity serves as the entry point for an app's interaction with the user. You implement an activity as a subclass of the Activity class.

An activity provides the window in which the app draws its UI. This window typically fills the screen, but may be smaller than the screen and float on top of other windows. Generally, one activity implements one screen in an app. For instance, one of an app's activities may implement a Preferences screen, while another activity implements a Select Photo screen.

Most apps contain multiple screens, which means they comprise multiple activities. Typically, one activity in an app is specified as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity in order

to perform different actions. For example, the main activity in a simple e-mail app may provide the screen that shows an e-mail inbox. From there, the main activity might launch other activities that provide screens for tasks like writing e-mails and opening individual e-mails.

Although activities work together to form a cohesive user experience in an app, each activity is only loosely bound to the other activities; there are usually minimal dependencies among the activities in an app. In fact, activities often start up activities belonging to other apps. For example, a browser app might launch the Share activity of a social-media app.

3.7.3 Speech Activity:

Speech Activity it is one of the activity in our emotion application. This activity contains various emotions in that activity like anger,fear. When we entering to our application it first start with splash screen and then after three seconds its going to the main activity. Now we will see lot of user interfaces like play button and some emotions list.

So when we click the play button in our application it automatically start wait to recognize the user speech. Once speech is completed user need to click the same button in that time buttons work like stop button. And then our application shows what is the emotion of your speech.

It recognize your speech and it will show the emoji based on your speech wave length for example your speech happy compare with other emotions and the emoji shows smiley emoji so this this application is very useful in healthcare sector to analyze the patients emotions.

4 SOFTWARE DEVELOPMENT METHODOLOGY:

4.1 SYSTEM REQUIREMENTS:

4.1.1 SOFTWARE SPECIFICATION:

- ❖ JDK
- ❖ Android Studio
- ❖ SQLite

4.1.2 HARDWARE SPECIFICATION:

- ❖ RAM : 8GB
- ❖ Hard disk : 80GB
- ❖ Processor : 2GHz

4.2 ANDROID:

A free, open source mobile platform. A Linux-based, multiprocessing, Multi threaded OS. Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

- ❖ Makes mobile development easy.
- ❖ Full phone software stack including applications
- ❖ Designed as a platform for software development
- ❖ Android is open source

- ❖ Android is free
- ❖ Community support

4.3 FEATURES:

Application framework enabling reuse and replacement of components Dalvik virtual machine optimized for mobile devices Integrated browser based on the open source Web Kit engine Optimized graphics powered by a custom 2D graphics library; 3Dgraphics based on the OpenGL ES 1.0 specification (hardware acceleration optional).SQLite for structured data.

4.4 STORAGE:

Media support for common audio, video, and still image formats(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)GSM Telephony (hardware dependent)Bluetooth, EDGE, 3G, and Wi-Fi (hardware dependent) Camera, GPS, compass, and accelerometer (hardware dependent) Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug in for the Android studio.

- ❖ Linux Kernel
- ❖ Android relies on Linux version 2.6 for core system services such as
- ❖ Security
- ❖ memory management
- ❖ process management
- ❖ network stack
- ❖ driver model

The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

4.5 ANDROID RUNTIME:

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Libraries Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

- ❖ System C Library
- ❖ Media Library
- ❖ Surface Manager
- ❖ Lib Web Core
- ❖ SGL
- ❖ 3D libraries
- ❖ Free Type
- ❖ SQLite

Being an open development platform, Android offers developers the ability to build extremely rich and innovative applications.

Developers have full access to the same framework APIs used by the core applications. Views – used to build applications (lists, grid, buttons, text boxes and even embedded web browser).

Content providers – enable applications to access data from other applications or share their own data.

Resource manager – provides access to non-code resources such as localized strings, graphic and layout files.

Notification manager – enables applications to display custom alerts in status bar.

Activity manager – manages life cycle of applications and provides navigation back stack.

Android ships with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others.

All applications are written using the Java programming language.

4.6 ARCHITECTURE:

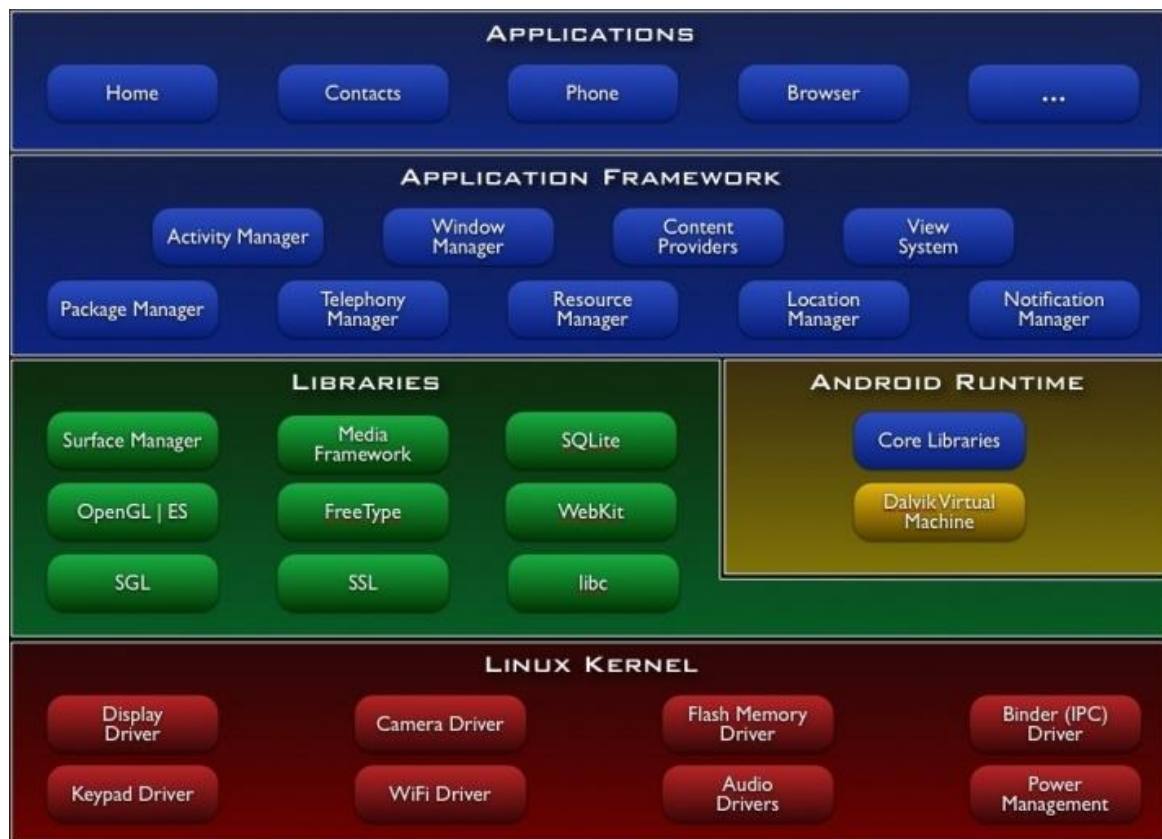


FIG: 4.6 ARCHITECTURE

4.6.1 ANATOMY OF AN ANDROID APPLICATION:

There are four building blocks for an Android application:

Activity - a single screen

Broadcast Receiver- to execute in reaction to an external event(Phone Ring)

Service - code that is long-lived and runs without a UI(Media Player)

Content Provider - an application's data to be shared with other applications.

4.7 ANDROID BUILDING BLOCKS:

These are the most important parts of the Android APIs:

Android Manifest.XML:

The control file-tells the system what to do with the top-level components

Activities:

An object that has a life cycle-is a chunk of code that does some work

Views:

An object that knows how to draw itself to the screen

Intent:

A simple message object that represents an "intention" to do something.

Notifications:

A small icon that appears in the status bar(SMS messages)for alerting the user.

Services:

A body of code that runs in the background.

Development Tools:

The Android SDK includes a variety of custom tools that help you develop mobile applications on the Android platform. Three of the most significant tools are:

Android Emulator:

A virtual mobile device that runs on our computer -use to design, debug, and test our applications in an actual Android run-time environment.

Android Development Tools:

Plug-in -for the Android studio - adds powerful extensions to the integrated environment.

Dalvik Debug Monitor Service (DDMS):

Integrated with Dalvik -this tool let us manage processes on an emulator and assists in debugging.

Android Asset Packaging Tool (AAPT):

Constructs the distributed Android package files (.apk)

Android Debug Bridge (ADB):

Provides link to a running emulator. Can copy files to emulator, install .apk files and run commands.

4.8 LIFECYCLE OF ACTIVITY:

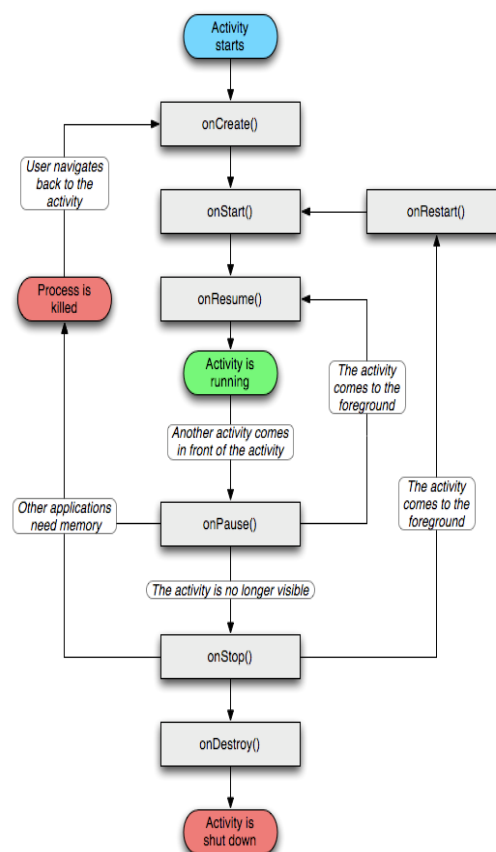


FIG:4.8 LIFECYCLE OF ACTIVITY

4.9 OVERVIEW OF XML:

XML (Extensible Mark-up Language) is a set of rules for encoding documents electronically. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

XML's design goals emphasize simplicity, generality, and usability over the Internet is a textual data format, with strong support via Unicode for the languages of the world. Although XML's design focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

4.9.1 Advantages:

The main usage of xml is we can store and retrieve data easily with the help of xml.

4.10 XML Introduction:

MIDP devices have memory constraints when it comes to code, both in terms of the amount of code you can store on the device, and memory available to applications at runtime. So, keeping the size of applications and features in check is of paramount importance to the J2ME developer. That's where small-sized XML parsers come into play.

4.10.1 XML parsers:

This section describes the XML parsing process and introduces some small XML parsers for MIDP.

4.10.2 XML parsing process:

The XML parsing process operates in three phases:

- ❖ XML input processing. In this stage, the application parses and validates the source document recognizes and searches for relevant information based on its location or its tagging in the source document; extracts the relevant information when it is located; and, optionally, maps and binds the retrieved information to business objects.

- ❖ Business logic handling. This is the stage in which the actual processing of the input information takes place. It might result in the generation of output information.
- ❖ XML output processing. In this stage, the application constructs a model of the document to be generated with the Document Object Model (DOM). It then either applies XSLT style sheets or directly serializes to XML.

An application that implements such a processing model is called an XML parser. You can integrate an XML parser into your Java applications with the Java API for XML Processing (JAXP). JAXP allows applications to parse and transform XML documents using an API that is independent of any particular XML processor implementation. Through a plug-in scheme, developers can change XML processor implementations without altering their applications.

4.11 Create Android Application:

The first step is to create a simple Android Application using Eclipse IDE. Follow the option File -> New -> Project Now name your application as Hello World using the wizard window as follows:

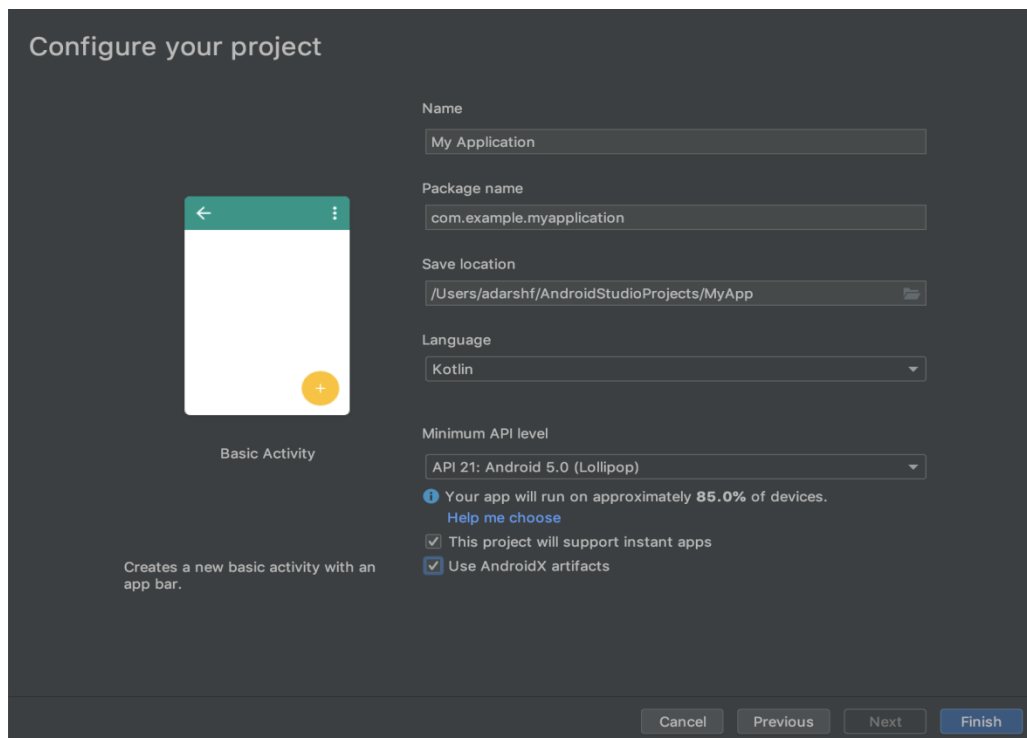


FIG: 4.11 CREATE ANDROID APPLICATION

4.12 ANDROID STUDIO:

On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- ❖ A flexible Gradle- based build system
- ❖ A fast and feature-rich emulator
- ❖ A unified environment where you can develop for all Android devices
- ❖ Apply Changes to push code and resource changes to your running app without restarting your app.
- ❖ Code templates and GitHub integration to help you build common app features and import sample code.
- ❖ Extensive testing tools and frameworks Link tools to catch performance, usability, version compatibility, and other problems.
- ❖ C++ and NDK support Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

4.12.1 TOOL WINDOWS:

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window. To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.

To return to the current default tool window layout, click Window > Restore Default Layout or customize your default layout by clicking Window > Store Current Layout as Default.

To show or hide the entire tool window bar, click the window icon in the bottom left-hand corner of the Android Studio window.

To locate a specific tool window, hover over the window icon and select the tool window from the menu.

4.13 JAVA PRPGRAMMING LANGUAGE:

4.13.1 History:

The JAVA language was created by James Gosling in June 1991 for use in a set top box project. The language was initially called Oak, after an oak tree that stood outside Gosling's office - and also went by the name Green - and ended up later being renamed to Java, from a list of random words. Gosling's goals were to implement a virtual machine and a language that had a familiar C/C++ style of notation. The first public implementation was Java 1.0 in 1995. It promised "Write Once, Run anywhere" (WORA), providing no-cost runtimes on popular platforms. It was fairly secure and its security was configurable, allowing network and file access to be restricted. Major web browsers soon incorporated the ability to run secure Java applets within web pages. Java quickly became popular. With the advent of Java 2, new versions had multiple configurations built for different types of platforms. For example, J2EE was for enterprise applications and the greatly stripped down version J2ME was for mobile applications. J2SE was the designation for the Standard Edition. In 2006, for marketing purposes, new J2 versions were renamed Java EE, Java ME, and Java SE, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a standard that is controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge although they were proprietary software. Sun's revenue from Java was generated by the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE) which is a subset of the SDK, the primary distinction being that in the JRE, the compiler, utility programs, and many necessary header files are not present.

On 13 November 2006, Sun released much of Java as free software under the terms of the GNU General Public License (GPL). On 8 May 2007 Sun finished the process, making Java's entire core code open source, aside from a small portion of code to which Sun did not hold the copyright.

4.13.2 Primary goals:

There were five primary goals in the creation of the Java language:

- It should use the object-oriented programming methodology.
- It should allow the same program to be executed on multiple operating systems.
- It should contain built-in support for using computer networks.
- It should be designed to execute code from remote sources securely.
- It should be easy to use by selecting what were considered the good parts of other object-oriented languages.

4.14 The Java Programming Language:

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- | | |
|--|---|
| ☐ Simple | ☐ Architecture neutral |
| ☐ Object oriented | ☐ Portable |
| ☐ Distributed | ☐ High performance |
| ☐ Multi thread | ☐ Robust |
| ☐ Dynamic | ☐ Secure |

Each of the preceding buzzwords is explained in The Java Language Environment , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the java compiler. A .class file does not contain code that is native to your processor; it instead contains byte codes — the machine language of the Java Virtual Machine¹ (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



FIG: 4.14 An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java Hot Spot virtual machine perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.

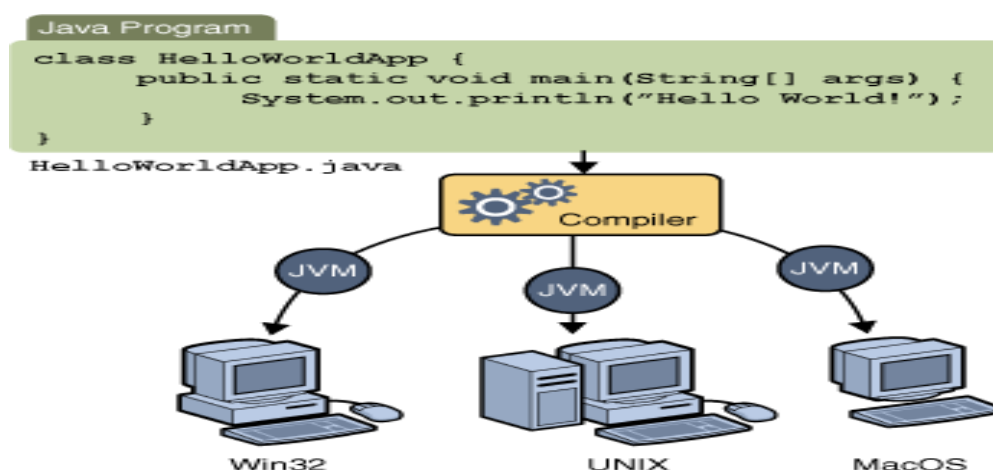


FIG: 4.14 Through the Java VM, the same application is capable of running on multiple platforms.

4.14.1 The Java Platform:

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- ❖ The Java Virtual Machine
- ❖ The Java Application Programming Interface (API)

You've already been introduced to the Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights some of the functionality provided by the API.

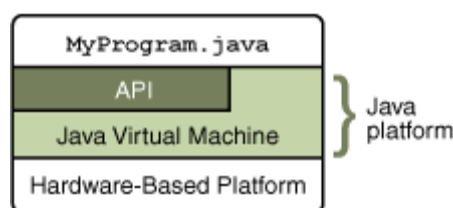


FIG: 4.14.1 The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

4.14.2 Java Runtime Environment:

The Java Runtime Environment, or JRE, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plugging. Sun also distributes a super set of the JRE called the Java 2 SDK (more commonly known as the JDK), which includes development tools such as the Java compiler, Java-doc, Jar and debugger.

One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in production, testing or development environments.

4.14.3 Uses of JAVA:

Blue is a smart card enabled with the secure, cross-platform, object-oriented Java Card API and technology. Blue contains an actual on-card processing chip, allowing for enhance able and multiple functionality within a single card. Applets that comply with the Java Card API specification can run on any third-party vendor card that provides the necessary Java Card Application Environment (JCAE). Not only can multiple applet programs run on a single card, but new applets and functionality can be added after the card is issued to the customer.

- ❖ Java Can is used in Chemistry.
- ❖ In NASA also Java is used.
- ❖ In 2D and 3D applications java is used.
- ❖ In Graphics Programming also Java is used.
- ❖ In Animations Java is used.
- ❖ In Online and Web Applications Java is used.

CHAPTER 5

RESULTS AND DISCUSSION:

Output:

5.1 Splash Screen:

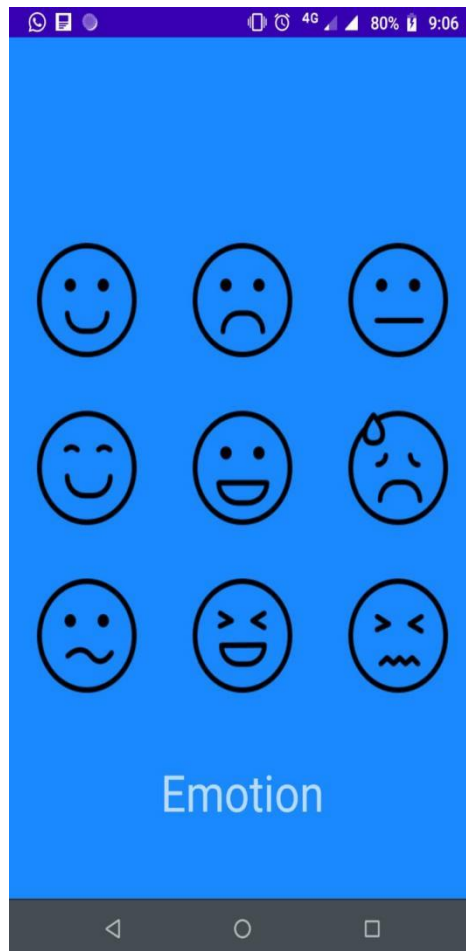


FIG: 5.1 SPLASH SCREEN

Android Splash Screen is the first screen visible to the user when the application's launched. Splash screen is one of the most vital screens in the application since it's the user's first experience with the application.

5.2 Main Screen:

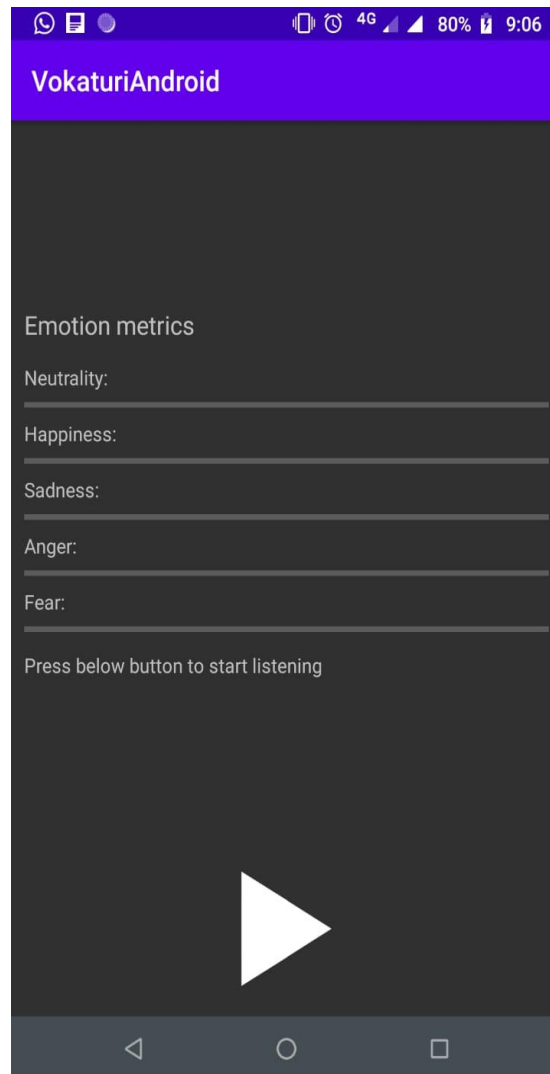


FIG: 5.2 MAIN SCREEN

The Activity class is designed to facilitate this paradigm. When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. In this way, the activity serves as the entry point for an app's interaction with the user. You implement an activity as a subclass of the Activity class.

5.3 Start Speech Screen:

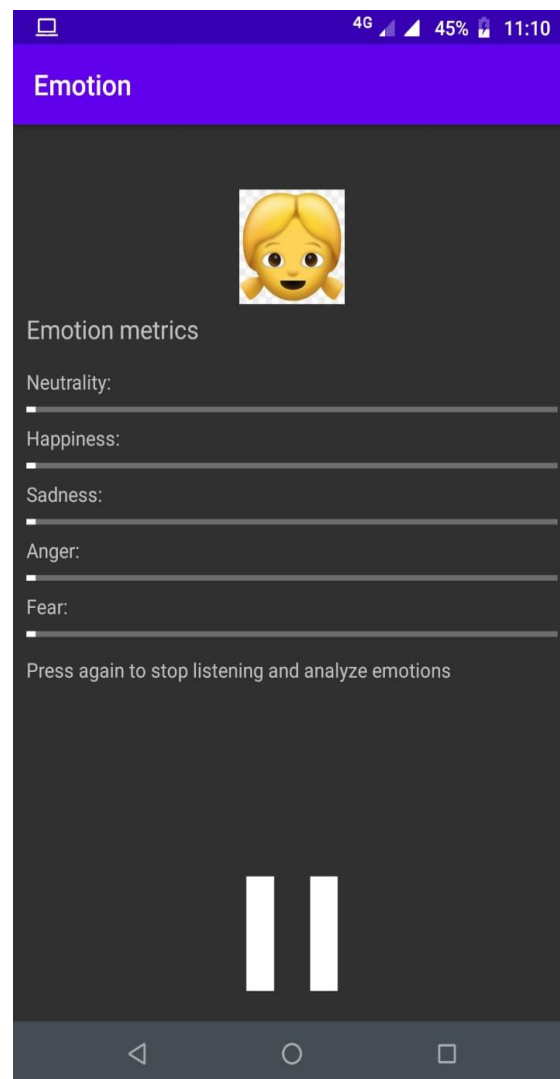


FIG: 5.3 STRAT SPEECH SCREEN

Speech Activity it is one of the activity in our emotion application. This activity contains various emotions in that activity like anger,fear. When we entering to our application it first start with splash screen and then after three seconds its going to the main activity. Now we will see lot of user interfaces like play button and some emotions list.

5.4 Fear Screen:

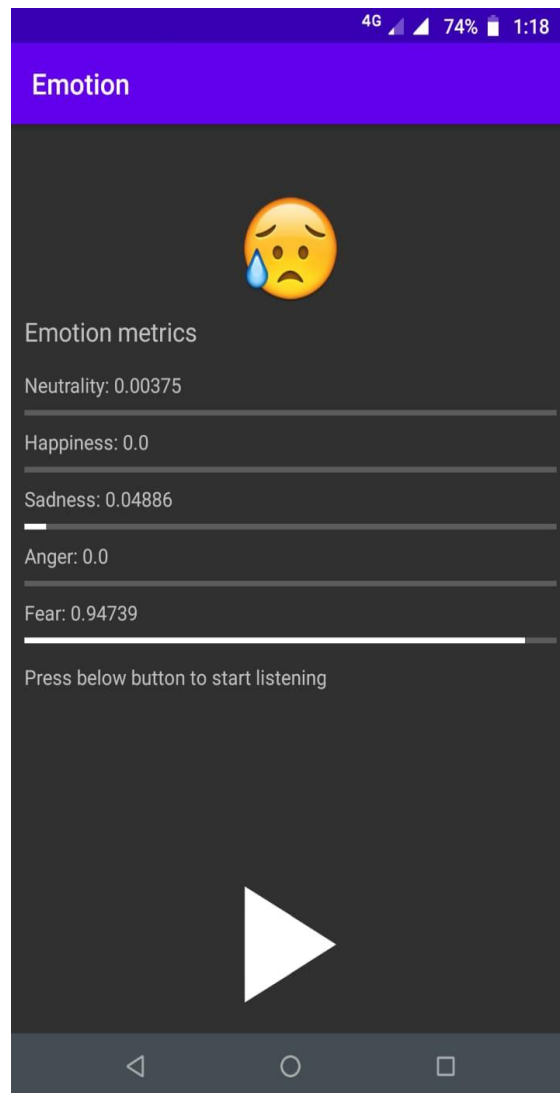


FIG: 5.4 FEAR SCREEN

5.5 Sadness Screen:

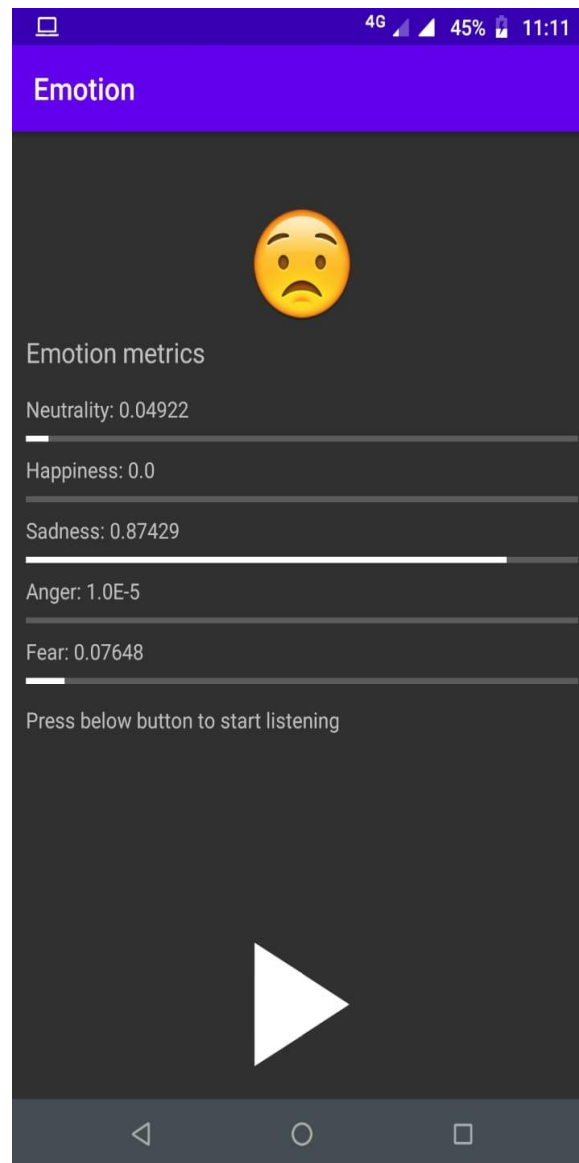


FIG: 5.5 SADNESS SCREEN

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION:

This paper proposed an effective emotion transplantation approach within an E2E-speech framework for generating expressive speech for a target speaker whose speech database includes only neutral style speech. By alternately updating a pre-trained expressive speech model in two directions, we not only generated the target speaker's voice characteristics but also successfully maintained the expressiveness characteristics of the pre-trained model. Thus, the proposed method successfully transplanted the ability to express appropriate emotions in the target speaker's voice. We verified the superior performance of the proposed approach through various quantitative and qualitative evaluations.

FUTURE WORK:

An effective emotion transplantation approach within an E2E-TTS framework for generating expressive speech. For a target speaker whose speech database includes only neutral style speech. By alternately updating a pre-trained expressive TTS model in two directions. we not only generated the target speaker's voice characteristics but also successfully maintained the expressiveness characteristics of the pre-trained model. In Healthcare sector and investigation to use them applications.

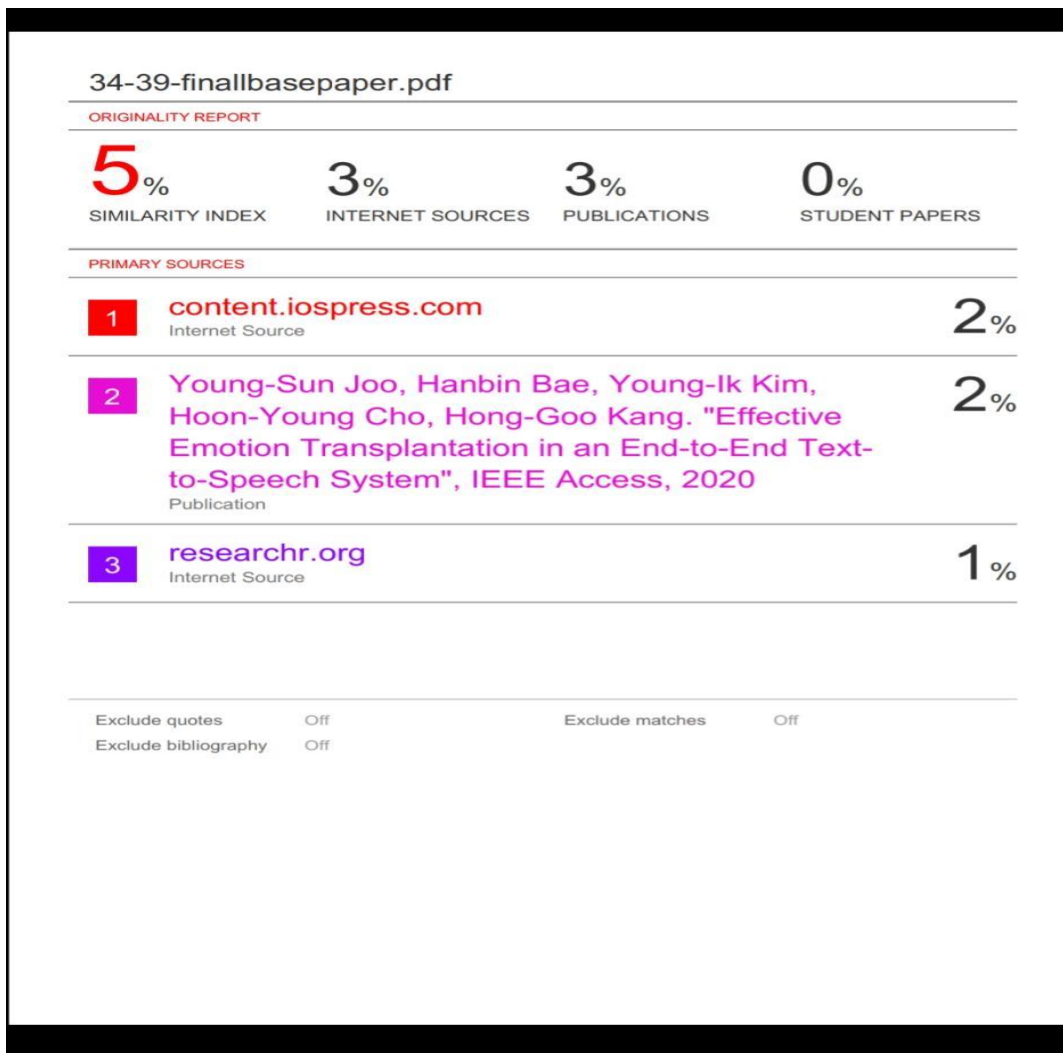
Thus, the method successfully transplanted the ability to express appropriate emotions in the target speaker's voice. We verified the superior performance approach on through various quantitative and qualitative evaluations. In this medical treatment like Psychology treatment should be using.

REFERENCE:

- [1] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in Proc. ICASSP, 2013, pp. 7962–7966.
- [2] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in Proc. Interspeech, 2014, pp. 1964–1968.
- [3] E. Song, F. K. Soong, and H.-G. Kang, "Effective Spectral and Excitation Modeling Techniques for LSTM-RNN-Based Speech Synthesis Systems," IEEE/ACM Trans. on Audio, Speech, and Language Processing, vol. 25, no. 11, pp. 2152–2161, 2017.
- [4] Z.-C. Liu, Z.-H. Ling, and L.-R. Dai, "Statistical Parametric Speech Synthesis Using Generalized Distillation Framework," IEEE Signal Processing Letters, vol. 25, no. 5, pp. 695–699, 2018.
- [5] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in Arxiv, 2016. [Online]. Available: <https://arxiv.org/abs/1609.03499>
- [6] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly et al., "Tacotron: Towards End-to-End Speech Synthesis," in Proc. Interspeech, 2017, pp. 4006–4010.
- [7] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang et al., "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning," in Proc. ICLR, 2018.
- [8] H. Tachibana and S. Aihara, "Efficiently Trainable Text-to-Speech System based on Deep Convolutional Networks with Guided Attention," in Proc. ICASSP, 2018, pp. 4784–4788.
- [9] L. Chen, N. Braunschweiler, and M. J. Gales, "Speaker and expression factorization for audiobook data: Expressiveness and transplantation," IEEE Trans. on Audio, Speech, and Language Processing, vol. 23, no. 4, pp. 605–618, 2015.

APPENDIX

A. PLAGIARISM REPORT



Effective Emotion Transplantation in an End to End Text to Speech System

¹ Kaviya
Department of Information Technology
Sathyabama Institute of Science and Technology,
Chennai, India
kaviyavijaykumar021099@gmail.com

¹ Jaya Rasika
Department of Information Technology
Sathyabama Institute of Science and Technology,
Chennai, India
rasikajaya29@gmail.com

Abstract— Within a text-to-speech (TTS) device that works from beginning to end, a successful strategy for transplanting the emotional expression in the case of a source speaker to the voice in a new target speaker. We adapt an expressive TTS model trained on an emotional speech database from a source speaker to represent a goal speaker's voice characteristics for whom only a database of neutral speech is open. To accomplish this, we developed two conditions for adaptation. One criteria is for reduce the difference in reconstruction between the reported and synthesised speech of the target speaker, so the idea that synthesised The voice features of the target speaker are present in speech. Another criteria to be reduce between the vectors of emotion embedding emotion loss derived dependent on the expressive voice as a model and the synthesised The goal speaker's expressive voice. In both quantitative and qualitative analyses, the proposed adaptation technique outperforms traditional methods.

Keywords—Speech; Emotion Recognition ;
Emoji ; Speech Expression ; Frequency

I. INTRODUCTION

Speech is one of the basic and natural way of communication among human beings. Emotions improves the expressiveness and effectiveness of expression. Various methods, such as laughing, shouting, and teasing and weeping. To be willing to recognition to understand emotions, one must first have a good understanding of what they are, which is a theory of emotions. Emotion theories have been around for a long time.

tradition in psychology, with elaborate models that can be used as the base for automatic emotion recognition. In the first section of this chapter, the most important ones in terms of speech are discussed. They're often thought about in terms of how feasible they are realize this is not the case with the more recent and complex theories in terms of practical implementations.

II. MODLES

The end to end speech system used in an emotion encoder that produces a degree of expressiveness. Emotion end to end speech systems which uses input and expressiveness to synthesise expressive speech conditions.

A. Emotion Recognition

We have combined the science of psychology , human speech to recognize that different emotions. Our speech analysis can identify five different type of emotional states in real time: Neutrality , Happiness, anger, Sadness and fear.

B. Speech Expression

A human being communicates his opinions, feelings, and emotions to others through expression. other. As a result, freedom of speech and expression are guaranteed it natural justice. Everyone has the right to freedom opinion as well as speech.

The Supreme Court has ruled that the terms "freedom of speech and expression" have a broad definition. The right to free speech and expression should be the ability to circulate the one's views through words is narrowly defined.

III. EMOTION TRANSPALATION APPROACH

Assume we have an individual who is expressive speech. The model was pre-trained with an expressive speech database from a source speaker, but the target speaker only has access to a neutral speech database. Adapting the pre-trained source is a clear concept speakers communicative speech model with the voice of the speaker who is being targeted neutral speech. Assume we have a pre-trained TTS model that is expressive an for a speaker who is the source, there is an database of expressive expression, but for the target speaker, there is Just a record of neutral expression. The adaption of the target speaker's pre-trained expressiveness is a clear concept speech model with the neutral voice of the target speaker.

A. END TO END SPEECH

E2E is a concept that refers to a number of business-to-speech we use the deep frameworks in our frameworks. convolutional speech framework due to its pace training alignment stability and pace. The speech The mel-spectrogram is then assumed by the audio decoder using autoregressive inference these combined embedded images. The spectrogram module is eventually transformed into a time-domain speech waveform.

when there are a lot of gaps in the focus pattern due to different speaking rates depending on the emotion groups.

The model's capacity to produce expressive speech is constrained since it is adapted to use neutral speech to recreate the target speaker's voice.

B. EQUATIONS

The encoder of emotions Memo as well as the TTS model MTTs are educated together. First and foremost, Memo S srcemo ref produces the embedding vector of emotion "e". Then, as input, e is passed to MTTs with name X.

$$a \square \square \square \square b \square \square \square \square \square \square$$

when there are a lot of gaps in the focus pattern due to different speaking rates depending on the emotion groups.

C. VOICE CHARACTERISTICS TO BE ADAPTED

- The E2E's network architectures and hyperparameters-speech module, as well as the module for encoding emotions.
- With the exception that the embedding vector for emotions and vectors for text embedding were concatenated. As a result, the audio embedding vectors' dimensions were modified to fit the concatenated embedding vector's.
- To assess Participants were asked to rate how close the speakers were how similar it is synthesised speech sounded to the voice of the speech.

IV. EXPERIMENT RESULT

Which implies a lot more distinct it was possible to synthesise expressive expression. emoTgtConv's emotion embedding vectors, on the other hand, revealed an unclear boundary between emotions. We determined the equivalent error rate (EER) for speakers using this process. verification and the emotion classification precision for the synthesised articulated speech of the target speaker samples: A lower EER suggests a higher speaker similarity, while a high EER indicates a low speaker similarity classification. The accuracy of the synthesised speech suggests that it faithfully communicates emotion in the input emotion embedding vector's expressiveness style. For the purpose of purpose of emotion classification we have a mission used the Bayesian naive classification a system. Emotion embedding vectors were used to obtain the mean and variance of each class are model parameters extracted derived the registered the speaker's expressive voice source speaker.

V. IDENTIFY THE ANALYSIS:

As a result, the audio embedding vectors' dimensions were modified to fit the embedding vector's concatenation. Mel-spectrograms are a form of spectrogram that is used to in 80 dimensions was taken out at

$$\square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square$$

Intervals between frames of 12.5 milliseconds and frame spans of 50 milliseconds from speech parts.

when working on the baseline expressive capacity speech when attempting to find out who the goal speaker is expressive when using the model, and 0.0001 when using the attempting to obtain the expressive model of the target speaker.

A. Figures and Tables

To assess we asked participants to rate the severity of the situation expressiveness of a voice signalling sent to the following information emotion use the following to create a mark five possible

1 answers = Totally different from the emotion label that has been annotated; 2 = In between the emotion label with annotations and 3 = Slightly expressive; 4=Somewhat expressive;5 = Extremely expressive is the annotated emotion label.

TABLE I. TABLE STYLES

Table	Table of Speech		
Head	Subhead of a table column	Loss calculation	Emotion
Sound	3.88 ±0.10	voice	Forzen

Table Head	Table of Speech		
Head	Table column subhead	Loss calculation	Emotion
Qualitative			

VI. CONCLUSION

An alternative was proposed in this article efficient transplantation of emotions method for producing For a target speaker with just a database of spoken words, expressive speech is needed contains neutral style speech within an E2E-speech

context. We don't just developed the speech characteristics of the target speaker, but also effectively preserved them by upgrading a pre-trained expressive model on a regular basis speech two-dimensional model.

References

- [1] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in Proc. ICASSP, 2013, pp. 7962–7966.
- [2] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in Proc. Interspeech, 2014, pp. 1964–1968.
- [3] E. Song, F. K. Soong, and H.-G. Kang, "Effective Spectral and Excitation Modeling Techniques for LSTM-RNN-Based Speech Synthesis Systems," IEEE/ACM Trans. on Audio, Speech, and Language Processing, vol. 25, no. 11, pp. 2152–2161, 2017.
- [4] Z.-C. Liu, Z.-H. Ling, and L.-R. Dai, "Statistical Parametric Speech Synthesis Using Generalized Distillation Framework," IEEE Signal Processing Letters, vol. 25, no. 5, pp. 695–699, 2018.
- [5] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in Arxiv, 2016. [Online]. Available: <https://arxiv.org/abs/1609.03499>
- [6] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly et al., "Tacotron: Towards End-to-End Speech Synthesis," in Proc. Interspeech, 2017, pp. 4006–4010.
- [7] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang et al., "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning," in Proc. ICLR, 2018.
- [8] H. Tachibana and S. Aihara, "Efficiently Trainable Text-to-Speech System based on Deep Convolutional Networks with Guided Attention," in Proc. ICASSP, 2018, pp. 4784–4788.

SOURCE CODE:

CODING:

activity_splash.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="@color/teal_200">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/i3"/>
        android:background="@color/teal_200"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Emotion"
        android:textSize="40dp"
        android:layout_marginTop="570dp"/>

        android:textColor="@color/black"/>
</RelativeLayout>
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    tools:context="com.projects.alshell.emotion.MainActivity">
```

```
<LinearLayout
```

```
    android:layout_marginTop="30dp"
```

```
    android:padding="10dp"
```

```
    android:orientation="vertical"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content">
```

```
<ImageView
```

```
    android:src="@drawable/emoji_default"
```

```
    android:layout_margin="5dp"
```

```
    android:id="@+id/emojiEmotionImageView"
```

```
    android:layout_gravity="center"
```

```
    android:scaleType="fitXY"
```

```
    android:layout_width="80dp"
```

```
    android:layout_height="80dp"/>
```

```
<TextView
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="Emotion metrics"
```

```
    android:layout_marginBottom="15dp"
```

```
    android:textSize="18dp"
```

<TextView

**android:text="Neutrality:"
android:id="@+id/textViewNeutrality"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>**

<ProgressBar

**style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:max="100"
android:id="@+id/progressBarNeutrality"
android:indeterminate="false"
android:progress="0"
android:layout_width="match_parent"
android:layout_height="20dp"/>**

<TextView

**android:text="Happiness:"
android:id="@+id/textViewHappiness"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>**

<ProgressBar

**style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:max="100"
android:id="@+id/progressBarHappiness"
android:indeterminate="false"
android:progress="0"
android:layout_width="match_parent"
android:layout_height="20dp"/>**

<TextView

**android:text="Sadness:"
android:id="@+id/textViewSadness"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>**

<ProgressBar

style="@style/Widget.AppCompat.ProgressBar.Horizontal"

```
android:max="100"
android:id="@+id/progressBarSadness"
android:indeterminate="false"
android:progress="0"
android:layout_width="match_parent"
android:layout_height="20dp"/>
```

<TextView

```
android:text="Anger:"
android:id="@+id/textViewAnger"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

<ProgressBar

```
style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:max="100"
android:id="@+id/progressBarAnger"
android:indeterminate="false"
android:progress="0"
android:layout_width="match_parent"
android:layout_height="20dp"/>
```

<TextView

```
android:text="Fear:"
android:id="@+id/textViewFear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

<ProgressBar

```
style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:max="100"
android:id="@+id/progressBarFear"
android:indeterminate="false"
android:progress="0"
android:layout_width="match_parent"
android:layout_height="20dp"/>
```

<TextView

```
        android:paddingTop="5dp"
        android:id="@+id/actionStatus"
        android:text="Press below button to start listening"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
```

```
</LinearLayout>
```

```
<com.projects.alshell.emotion.PlayPauseButton
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:layout_margin="20dp"
    android:id="@+id/playPauseButton"
    android:layout_width="80dp"
    android:layout_height="80dp"/>
```

```
</RelativeLayout>
```

Activity_splash.java:

```
package com.projects.alshell.emotion;
```

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
```

```
public class Activity_Splash extends Activity {
    Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_splash);
```

```
handler = new Handler();
```

```
handler.postDelayed(new Runnable() {
```

```
    @Override
```

```
    public void run() {
```

```
        Intent intent = new Intent(Activity_Splash.this,MainActivity.class);
```

```
        startActivity(intent);
```

```
        finish();
```

```
    }
```

```
},3000);
```

```
}
```

```
}
```

MainActivity.java:

```
package com.projects.alshell.emotion;
```

```
import android.annotation.SuppressLint;
```

```
import android.content.pm.PackageManager;
```

```
import android.os.Bundle;
```

```
import android.support.annotation.NonNull;
```

```
import android.support.v4.app.ActivityCompat;
```

```
import android.support.v7.app.AppCompatActivity;
```

```
import android.view.View;
```

```
import android.widget.ImageView;
```

```
import android.widget.ProgressBar;
```

```
import android.widget.TextView;
```

```
import android.widget.Toast;
```

```
import com.projects.alshell.vokaturi.Emotion;
```

```
import com.projects.alshell.vokaturi.EmotionProbabilities;
```

```
import com.projects.alshell.vokaturi.Vokaturi;
```

```
import com.projects.alshell.vokaturi.VokaturiException;
```

```
import static android.Manifest.permission.RECORD_AUDIO;
```

```
import static android.Manifest.permission.WRITE_EXTERNAL_STORAGE;
import static com.projects.alshell.vokaturi.Vokaturi.logD;
```

```
@SuppressWarnings("deprecation")
```

```
public class MainActivity extends AppCompatActivity
{
```

```
    private static final int PERMISSIONS_REQUEST_CODE = 5;
```

```
    private ImageView emojiEmotionImageView;
```

```
    private ProgressBar progressBarNeutrality;
```

```
    private ProgressBar progressBarHappiness;
```

```
    private ProgressBar progressBarSadness;
```

```
    private ProgressBar progressBarAnger;
```

```
    private ProgressBar progressBarFear;
```

```
    private TextView textViewNeutrality;
```

```
    private TextView textViewHappiness;
```

```
    private TextView textViewSadness;
```

```
    private TextView textViewAnger;
```

```
    private TextView textViewFear;
```

```
    private TextView actionStatus;
```

```
    private PlayPauseButton playPauseButton;
```

```
    private Vokaturi vokaturiApi;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState)
```

```
{
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```

try {
    logD("About to instantiate the library");
    vokaturiApi = Vokaturi.getInstance(MainActivity.this);
} catch (VokaturiException e) {
    e.printStackTrace();
}

```

```

        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{RECORD_AUDIO, WRITE_EXTERNAL_STORAGE},
PERMISSIONS_REQUEST_CODE);

```

```

        initializeViews();

        setListeners();
    }

```

```

private void initializeViews()
{
    emojiEmotionImageView = (ImageView)
findViewById(R.id.emojiEmotionImageView);
    progressBarNeutrality = (ProgressBar)
findViewById(R.id.progressBarNeutrality);
    progressBarHappiness = (ProgressBar)
findViewById(R.id.progressBarHappiness);
    progressBarSadness = (ProgressBar)
findViewById(R.id.progressBarSadness);
    progressBarAnger = (ProgressBar) findViewById(R.id.progressBarAnger);
    progressBarFear = (ProgressBar) findViewById(R.id.progressBarFear);
    playPauseButton = (PlayPauseButton) findViewById(R.id.playPauseButton);

    textViewNeutrality = (TextView) findViewById(R.id.textViewNeutrality);
    textViewHappiness = (TextView) findViewById(R.id.textViewHappiness);
    textViewSadness = (TextView) findViewById(R.id.textViewSadness);
    textViewAnger = (TextView) findViewById(R.id.textViewAnger);
}

```

```

        textViewFear = (TextView) findViewById(R.id.textViewFear);

        actionStatus = (TextView) findViewById(R.id.actionStatus);

    }

    private void setListeners()
    {
        playPauseButton.setOnControlStatusChangeListener(new
        PlayPauseButton.OnControlStatusChangeListener()
        {
            @Override
            public void onStatusChange(View view, boolean state)
            {
                if(state)
                {
                    startListening();
                } else {
                    stopListening();
                }
            }
        });
    }

    @SuppressWarnings("SetTextI18n")
    private void startListening()
    {
        if(vokaturiApi != null)
        {
            try {
                setListeningUI();
                vokaturiApi.startListeningForSpeech();
            } catch (VokaturiException e) {

```

```

        setNotListeningUI();
        if(e.getErrorCode() ==
VokaturiException.VOKATURI_DENIED_PERMISSIONS)
        {
            Toast.makeText(this, "Grant Microphone and Storage
permissions in the app settings to proceed", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(this, "There was some problem to start
listening audio", Toast.LENGTH_SHORT).show();
        }
    } catch (IllegalStateException e)
    {
        setNotListeningUI();
        e.printStackTrace();
    }
}

private void setListeningUI()
{
    actionStatus.setText("Press again to stop listening and analyze emotions");
    progressBarNeutrality.setIndeterminate(true);
    progressBarHappiness.setIndeterminate(true);
    progressBarSadness.setIndeterminate(true);
    progressBarAnger.setIndeterminate(true);
    progressBarFear.setIndeterminate(true);

    emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.girl));
}

@SuppressLint("SetTextI18n")
private void stopListening()
{

```

```

    if(vokaturiApi != null)
    {
        setNotListeningUI();

        try {
            showMetrics(vokaturiApi.stopListeningAndAnalyze());
        } catch (VokaturiException e) {
            if(e.getErrorCode() ==
VokaturiException.VOKATURI_NOT_ENOUGH_SONORANCY)
            {
                Toast.makeText(this, "Please speak a more clear and avoid
noise around your environment", Toast.LENGTH_LONG).show();
            }
        } catch (IllegalStateException e)
        {
            e.printStackTrace();
        }
    }
}

private void setNotListeningUI()
{
    actionStatus.setText("Press below button to start listening");
    progressBarNeutrality.setIndeterminate(false);
    progressBarHappiness.setIndeterminate(false);
    progressBarSadness.setIndeterminate(false);
    progressBarAnger.setIndeterminate(false);
    progressBarFear.setIndeterminate(false);
}

@SuppressLint("SetTextI18n")
private void showMetrics(EmotionProbabilities emotionProbabilities)
{

```

```

emotionProbabilities.scaledValues(5);

logD("showMetrics for, " + emotionProbabilities.toString());
textViewNeutrality.setText("Neutrality: " + emotionProbabilities.Neutrality);
textViewHappiness.setText("Happiness: " + emotionProbabilities.Happiness);
textViewSadness.setText("Sadness: " + emotionProbabilities.Sadness);
textViewAnger.setText("Anger: " + emotionProbabilities.Anger);
textViewFear.setText("Fear: " + emotionProbabilities.Fear);

showEmojiBasedOnMetrics(emotionProbabilities);

progressBarNeutrality.setProgress(normalizeForProgressBar(emotionProbabilities.Neutrality));

progressBarHappiness.setProgress(normalizeForProgressBar(emotionProbabilities.Happiness));

progressBarSadness.setProgress(normalizeForProgressBar(emotionProbabilities.Sadness));

progressBarAnger.setProgress(normalizeForProgressBar(emotionProbabilities.Anger));

progressBarFear.setProgress(normalizeForProgressBar(emotionProbabilities.Fear));

}

private int normalizeForProgressBar(double val)
{
    if(val < 1)
    {
        return (int)(val * 100);
    } else {

```

```

        return (int)(val * 10);
    }
}

private void showEmojiBasedOnMetrics(EmotionProbabilities
emotionProbabilities)
{
    Emotion capturedEmotion = Vokaturi.extractEmotion(emotionProbabilities);
    if(capturedEmotion == Emotion.Neutral)
    {

emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.emoji_neutral));
    } else if(capturedEmotion == Emotion.Happy)
    {

emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.emoji_happiness));
    }
    else if(capturedEmotion == Emotion.Sad)
    {

emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.emoji_sadness));
    }
    else if(capturedEmotion == Emotion.Angry)
    {

emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.emoji_anger));
    }
    else if(capturedEmotion == Emotion.Feared)
    {

```

```

emojiEmotionImageView.setImageDrawable(getResources().getDrawable(R.drawable
.emoji_fear));
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
                                     @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

    if (requestCode == PERMISSIONS_REQUEST_CODE)
    {
        if(grantResults[0] != PackageManager.PERMISSION_GRANTED)
        {
            Toast.makeText(this, "Audio recording permissions denied.",
Toast.LENGTH_SHORT).show();
        }
    }

}

}

```

PlayPauseButton.java:

```
package com.projects.alshell.emotion;
```

```

import android.animation.ValueAnimator;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

```

```

import android.graphics.Path;
import android.graphics.drawable.Drawable;
import android.os.Parcel;
import android.os.Parcelable;
import android.support.annotation.NonNull;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

```

```

public class PlayPauseButton extends View

```

```

{

```

```

    private final static double SQRT_3 = Math.sqrt(3);

```

```

    private final static int SPEED = 1;

```

```

    private final Point mPoint;

```

```

    private Paint mPaint;

```

```

    private Path mLeftPath;

```

```

    private Path mRightPath;

```

```

    private ValueAnimator mCenterEdgeAnimator;

```

```

    private ValueAnimator mLeftEdgeAnimator;

```

```

    private ValueAnimator mRightEdgeAnimator;

```

```

    private boolean mPlayed;

```

```

    private int mBackgroundColor = Color.WHITE;

```

```

    private ValueAnimator.AnimatorUpdateListener mAnimatorUpdateListener =

```

```

        new ValueAnimator.AnimatorUpdateListener() {

```

```

            @Override public void onAnimationUpdate(ValueAnimator
valueAnimator) {

```

```

                invalidate();

```

```

            }

```

```

        };

```

```

    private OnControlStatusChangeListener mListener;

```

```

public PlayPauseButton(Context context) {
    this(context, null, 0);
}

```

```

public PlayPauseButton(Context context, AttributeSet attrs) {
    this(context, attrs, 0);
}

```

```

public PlayPauseButton(Context context, AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);

    mPoint = new Point();
    initView();
}

```

```

private void initView() {
    setUpPaint();
    setUpPath();
    setUpAnimator();
}

```

```

private void setUpAnimator() {
    if (mPlayed) {
        mCenterEdgeAnimator = ValueAnimator.ofFloat(1.f, 1.f);
        mLeftEdgeAnimator = ValueAnimator.ofFloat((float) (-0.2f * SQRT_3),
(float) (-0.2f * SQRT_3));
        mRightEdgeAnimator = ValueAnimator.ofFloat(1.f, 1.f);
    } else {
        mCenterEdgeAnimator = ValueAnimator.ofFloat(0.5f, 0.5f);
        mLeftEdgeAnimator = ValueAnimator.ofFloat(0.f, 0.f);
        mRightEdgeAnimator = ValueAnimator.ofFloat(0.f, 0.f);
    }
}

```

```

        mCenterEdgeAnimator.start();
        mLeftEdgeAnimator.start();
        mRightEdgeAnimator.start();
    }

```

```

private void setUpPaint() {
    mPaint = new Paint();
    mPaint.setColor(mBackgroundColor);
    mPaint.setAntiAlias(true);
    mPaint.setStyle(Paint.Style.FILL);
}

```

```

private void setUpPath() {
    mLeftPath = new Path();
    mRightPath = new Path();
}

```

```

@Override protected void onDraw(Canvas canvas) {
    mPoint.setHeight(canvas.getHeight());
    mPoint.setWidth(canvas.getWidth());

    mLeftPath.reset();
    mRightPath.reset();

    mLeftPath.moveTo(mPoint.getX(-0.5 * SQRT_3), mPoint.getY(1.f));
    mLeftPath.lineTo(mPoint.getY((Float)
mLeftEdgeAnimator.getAnimatedValue()) + 0.7f,
                    mPoint.getY((Float) mCenterEdgeAnimator.getAnimatedValue()));
    mLeftPath.lineTo(mPoint.getY((Float)
mLeftEdgeAnimator.getAnimatedValue()) + 0.7f,

```

```

        mPoint.getY(-1* (Float)
mCenterEdgeAnimator.getAnimatedValue()));
        mLeftPath.lineTo(mPoint.getX(-0.5 * SQRT_3), mPoint.getY(-1.f));

        mRightPath.moveTo(mPoint.getY(-1*(Float)
mLeftEdgeAnimator.getAnimatedValue()),
        mPoint.getY((Float) mCenterEdgeAnimator.getAnimatedValue()));
        mRightPath.lineTo(mPoint.getX(0.5 * SQRT_3),
        mPoint.getY((Float) mRightEdgeAnimator.getAnimatedValue()));
        mRightPath.lineTo(mPoint.getX(0.5 * SQRT_3),
        mPoint.getY(-1 * (Float) mRightEdgeAnimator.getAnimatedValue()));
        mRightPath.lineTo(mPoint.getY(-1(Float)
mLeftEdgeAnimator.getAnimatedValue()),
        mPoint.getY(-1* (Float)
mCenterEdgeAnimator.getAnimatedValue()));

        canvas.drawPath(mLeftPath, mPaint);
        canvas.drawPath(mRightPath, mPaint);
    }

```

```

public void startAnimation() {
    mCenterEdgeAnimator = ValueAnimator.ofFloat(1.f, 0.5f);
    mCenterEdgeAnimator.setDuration(100 * SPEED);
    mCenterEdgeAnimator.addUpdateListener(mAnimatorUpdateListener);

    mLeftEdgeAnimator = ValueAnimator.ofFloat((float) (-0.2 * SQRT_3), 0.f);
    mLeftEdgeAnimator.setDuration(100 * SPEED);
    mLeftEdgeAnimator.addUpdateListener(mAnimatorUpdateListener);

    mRightEdgeAnimator = ValueAnimator.ofFloat(1.f, 0.f);
    mRightEdgeAnimator.setDuration(150 * SPEED);
    mRightEdgeAnimator.addUpdateListener(mAnimatorUpdateListener);
}

```

```

    if (!mPlayed) {
        mCenterEdgeAnimator.start();
        mLeftEdgeAnimator.start();
        mRightEdgeAnimator.start();
    } else {
        mCenterEdgeAnimator.reverse();
        mLeftEdgeAnimator.reverse();
        mRightEdgeAnimator.reverse();
    }
}

```

```

public void setOnControlStatusChangeListener(OnControlStatusChangeListener
listener) {
    mListener = listener;
}

```

```

@Override public boolean onTouchEvent(@NonNull MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            setPlayed(!mPlayed);
            startAnimation();
            if (mListener != null) {
                mListener.onStatusChange(this, mPlayed);
            }
            break;
    }
    return false;
}

```

```

@Override public Parcelable onSaveInstanceState() {
    Parcelable superState = super.onSaveInstanceState();
    SavedState savedState = new SavedState(superState);
}

```

```
        savedState.played = isPlayed();  
        return savedState;  
    }
```

```
@Override public void onRestoreInstanceState(Parcelable state) {  
    SavedState savedState = (SavedState) state;  
    super.onRestoreInstanceState(savedState.getSuperState());  
    setPlayed(savedState.played);  
    setUpAnimator();  
    invalidate();  
}
```

```
@Override public void setBackground(Drawable background) {  
}
```

```
public boolean isPlayed() {  
    return mPlayed;  
}
```

```
public void setPlayed(boolean played) {  
    if (mPlayed != played) {  
        mPlayed = played;  
        invalidate();  
    }  
}
```

```
public void setColor(int color) {  
    mBackgroundColor = color;  
    mPaint.setColor(mBackgroundColor);  
}
```

```

        invalidate();
    }

    static class SavedState extends BaseSavedState {

        public static final Parcelable.Creator<SavedState> CREATOR =
            new Parcelable.Creator<SavedState>() {
                public SavedState createFromParcel(Parcel in) {
                    return new SavedState(in);
                }

                public SavedState[] newArray(int size) {
                    return new SavedState[size];
                }
            };

        boolean played;

        SavedState(Parcelable superState) {
            super(superState);
        }

        private SavedState(Parcel in) {
            super(in);
            played = (Boolean) in.readValue(null);
        }

        @Override public void writeToParcel(@NonNull Parcel out, int flags) {
            super.writeToParcel(out, flags);
            out.writeValue(played);
        }
    }

```

```

static class Point {

    private int width;

    private int height;

    public void setWidth(int width) {
        this.width = width;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public float getX(float x) {
        return (width / 2) * (x + 1);
    }

    public float getY(float y) {
        return (height / 2) * (y + 1);
    }

    public float getX(double x) {
        return getX((float) x);
    }

    public float getY(double y) {
        return getY((float) y);
    }
}

public interface OnControlStatusChangeListener {
    void onStatusChange(View view, boolean state);
}

```