BI-DIRECTIONAL AUGMENTATIVE & ALTERNATIVE COMMUNICATION SYSTEM FOR THE SPEECH & HEARING IMPAIRED

A report submitted to

RAMAIAH INSTITUTE OF TECHNOLOGY

Bengaluru

ISP SENIOR PROJECT

as partial fulfillment of the requirement for

Bachelor of Engineering (B.E) in Information Science and Engineering

by

ABHISHEK R MISHRA	**(USN- 1MS16IS003)**
SIVA NANDAN REDDY RENDEDDULA	**(USN- 1MS16IS097)**
VISHAL CHAVAN	**(USN- 1MS16IS115)**
SANDHYA C	**(USN- 1MS16IS135)**

under the guidance of
Dr. LINGARAJU GM

RAMAIAH

Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY
May 2020

Department of Information Science and Engineering

Ramaiah Institute of Technology

Bengaluru – 54

RAMAIAH

Institute of Technology

CERTIFICATE

This is to certify that ABHISHEK R MISHRA (USN- 1MS16IS003), SIVA NANDAN REDDY RENDEDDULA (USN- 1MS16IS097), VISHAL CHAVAN (USN- 1MS16IS115) AND SANDHYA C (USN- 1MS16IS135) who were working for their IS821 SENIOR PROJECT under my guidance, have completed the work as per my satisfaction with the topic BI-DIRECTIONAL AUGMENTATIVE AND ALTERNATIVE COMMUNICATION SYSTEM FOR THE SPEECH AND HEARING IMPAIRED. To the best of my understanding the work to be submitted in dissertation does not contain any work, which has been previously carried out by others and submitted by the candidates for themselves for the award of any degree anywhere.

(Guide)	(Head of the Department)
Dr. Lingaraju G M	Dr. Vijay Kumar B P
Professor	Professor & Head,
Dept. of ISE	Dept. of ISE

(Examiner 1) (Examiner 2)

Name

Signature

Department of Information Science and Engineering
Ramaiah Institute of Technology

Bengaluru - 54

RAMAIAH

Institute of Technology

DECLARATION

We hereby declare that the entire work embodied in this IS821 SENIOR PROJECT report has been carried out by us at Ramaiah Institute of Technology under the supervision of **Dr. Lingaraju G M**. This project report has not been submitted in part or full for the award of any diploma or degree of this or any other University.

ABHISHEK R MISHRA	(USN- 1MS16IS003)
SIVA NANDAN REDDY RENDEDDULA	(USN- 1MS16IS097)
VISHAL CHAVAN	(USN- 1MS16IS115)
SANDHYA C	(USN- 1MS16IS135)

Acknowledgements

First and foremost we would like to express our immense gratitude to Dr. Lingaraju G. M, Department Of Information Science, MSRIT whose sincerity and encouragement we will never forget. He has been our support as we overcame all our difficulties in the completion of the project.

We wish to express our sincere thanks to Dr. Vijaya Kumar BP, Professor and HOD of Information Science & Engineering who guided and motivated us to complete this project. We would like to thank him for providing us with all necessary resources for this project.

We would also like to thank our Principal Dr. NVR Naidu for his support and encouragement. This work would not have been completed without the invaluable help provided by the other teaching and non-teaching staff members of the ISE Department.

All sentences and passages quoted in this report from other people’s work have been specifically acknowledged by clear cross-referencing to author, work and pages(s). Any illustrations which are not the work of the author of this report have been used (where possible) with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Abstract

Sign language are languages that use manual/visual tools such as hand gestures, facial expressions and body movements as a method of communication. These fully-fledged languages are independent of spoken languages and are the primary form of communication between disabled persons. Different regions practice different versions of sign language equivalent of English, these include American Sign Language(ASL), British Sign Language, Indian Sign Language(ISL). These constraints cause a major barrier in communication.

The proposed application focuses on providing a bi-directional communication tool for smooth and real-time communication between an abled and disabled person on a mobile platform. The application uses a Convolution Neural Network(CNN) model to translate hand gestures to their equivalent English alphabet and display it. It also utilizes a Speech-to-Text API to translate spoken language to text.

Table of Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

1. Introduction

1.1 Motivation

Consequences of hearing impairment lead to inability to understand speech sound, decreased capability to communicate, delay in language development, economic and educational backwardness, social isolation and stigmatization. It is the most frequent sensory deficit in the human population, affecting more than 360 million people in the world. This constitutes a substantial 5.3% of the world's population. The prevalence and incidence of hearing impairment in India also are substantially high. The high burden of deafness globally and in India is largely preventable and avoidable. The prevalence of deafness in South-East Asia ranges from 4.6% to 8.8%.

The estimated prevalence of adult-onset deafness in India is 7.6% and childhood-onset deafness is 2%. According to a study by the World Health Organization, five million Indian children were suffering from hearing and speech impairment in 2016.

1.2 Constraints and Requirements

A) **Dataset and Data Preprocessing -** There are ISL Dataset of character hand signs available. As a result a dataset had to be created from scratch.

B) **Capturing the video (OpenCV) -** Multiple Image Preprocessing techniques were required to filter out the skin.

C) **Analysis of the gestures (CNN using Keras) –** Hyper-parameter tuning. Dedicated GPU for processing was a necessity as we used trial and error methods.

D) **Translation to Text -** English words datasets for word formation and prediction required.

E) **Text to Speech (API) -** Python module was required.

F) **Speech to Text (API) -** Google API was required

G) **Rapid switching between the modes for two way communication -** Latency when switching between modes and ease of use.

H) **Integration of the application on a mobile platform -** Use of Android Studio or any other 3rd party application to port the code to a standalone application.

1.3 Problem Statement

Speech impairment is a disability which affects one's ability to speak and hear. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people. The aim of the project is to bridge this gap in communication and make it easier for disabled to communicate more freely.

1.4 Scope and Objectives

1.4.1. Scope

A) **Provides two different modes -** Gesture Translation and Speech translation Mode(s) of operation.

B) **Application interface to enable smooth communication between an abled and a disabled person (speech and hearing disability) -** UI to allow ease of access to the user and smooth transitioning between modes, in real time.

C) **One mode captures video and processes the same** - This recognizes the hand gestures and translates them to text, which will be output as speech.

D) **Uses words** - They are spelled out letter by letter.

E) **Second mode** - This converts speech to text for the disabled person to view.

1.4.2 Objectives

A) **Create a Database of generic gesture symbols used in Sign Language Communication -** A dataset of Indian Sign Language needs to be created for use, consisting of 6000+ images.

B) **Pattern Recognition in real time, to recognize the standard symbols (pattern) using the camera as an input .Implementing existing ML algorithms or tools as a part of the system. Outcome of this module is word (text) -** The hand sign is read by the webcam (or) camera input and the appropriate character for the sign is relayed to the user. Use of Convolution Neural Networks (CNN) to train the model on the dataset.

C) **Doing word-sentence formation and prediction and presenting the result to the users for validation -** Words are formed character by character and then chained to form a sentence.

D) **Building Speech to Text and Text to Speech Module -** Use of python API's to do the text-to-speech translation and vice-versa.

1.5 Proposed Model

The proposed model has the following major steps:

A) Data Collection from camera of hand signs of various individuals.
B) Image pre-processing using multiple techniques such as resizing, skin filtering etc.
C) Gesture Translation and Mapping leading to formation of words.
D) Speech-to-Text translation.
E) UI to facilitate operation of model.

1.6 Organization of Report

In order to explain the developed system, the following sections are covered :

- **Literature Review :** describes the study of existing AAC systems and techniques taken into account prior to development of this system.
- **System Analysis and Design :** provides a detailed walkthrough of the software engineering methodology adopted to implement along with an overview of the system modules incorporated into the system.
- **Modeling and Implementation :** provides a deeper insight into the working of the model. The various modules and their interactions are depicted with diagrams.
- **Testing :** the model to ensure bug/error free model along with the **Results** obtained. **Discussion** gives detailed analysis on Quality.
- **Conclusion :** about the Results obtained and **Future Scope** of the model is highlighted.

Chapter 2
Literature Review

2. Literature Review

[1] Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil’s research paper Sign Language Interpreter using Image Processing and Machine Learning (2019) proposes an application that acquires image data using the webcam of the computer, then it is preprocessed using a combinational algorithm and recognition is done using template matching. There remains a challenge for people who do not understand sign language to communicate with speech impaired people.

[2] Kshitij Bantupalli, Ying Xie’s research paper American Sign Language Recognition Using Machine Learning and Computer Vision (2019) We propose to use a CNN (Convolutional Neural Network) named Inception to extract spatial features from the video stream for Sign Language Recognition (SLR). Then by using a LSTM (Long Short-Term Memory), an RNN (Recurrent Neural Network) model, we can extract temporal features from the video sequences. While testing with different skin tones, the model dropped accuracy if it hadn’t been trained on a certain skin tone was made to predict on it. The model also suffered from loss of accuracy with the inclusion of faces.

[3] Abhijith Bhaskaran K, Anoop G Nair, Deepak Ram K, Krishnan Ananthanarayanan, H R Nandi Vardhan’s research paper Smart Gloves for Hand Gesture Recognition (Sign Language to Speech Conversion System) (2016) proposed a smart glove which can convert sign language to speech output. The glove is embedded with flex sensors and an IMU to recognize the gesture. The paper recognizes a few limitations - There is also scope for improvement in the system by incorporating contact sensors. Data Glove is not easily obtainable.

[4] Radici, E., Bonacina, S., & De Leo, G’s paper Design and development of an AAC app based on a speech-to-symbol technology (2016) proposes To present design and development of AAC app that uses speech to symbol technology to model language. App is intended to be adopted by communication partners who want to engage in interventions focused on improving communication skills. Our app goal is translating simple speech sentences into a set of symbols that are understandable by children with needs. The paper found a limitation that participants reported that teaching parents how to use AAC devices was a barrier.

[5] Rajaganapathy. S, Aravind. B, Keerthana. B, Sivagami. M's research paper Conversation of Sign Language to Speech with Human Gestures (2015) proposes to convert the human sign language to Voice with human gesture understanding and motion capture, with the help of Microsoft Kinect. There are a few systems available for sign language to speech conversion but none of them provide natural user interface. Due to use of sophisticated external technology the paper faced quite a few limitations some of which are- Sensor has an optimal range between 40 cm - 4M, it cannot recognize human objects beyond this. Another limitation is the NLP system that interprets the generated text to speech. The gesture tracking is also limited only to 2 individuals.

[6] Hongwu Yang, Xiaochun An, Dong Pei, Yitong Liu's research paper Towards realizing gesture-to-speech conversion with a HMM-based bilingual speech synthesis system (2015) proposes An improved speeded up robust features algorithm is adopted for static gesture recognition by combining Kinect sensor. Hidden Markov Model based Mandarin -Tibetan bilingual speech synthesis system is developed by using speaker adaptive training. The paper faced a few limitations - The model is strictly restricted to the languages Mandarin and Tibetan. A Kinect Sensor is expensive and not portable. It is not readily accessible to everyone.

[7] V. Padmanabhan, M. Sornalatha's paper Hand gesture recognition and voice conversion system for dumb people published in International Journal of Scientific & Engineering Research, Volume 5, Issue 5, May-2014 ISSN 2229-5518 (2016) proposes a new technique called artificial speaking mouth for dumb people. Messages are kept in a database. In the real time the template database is fed into a microcontroller and the motion sensor is fixed in their hand. The output of the system is using the speaker. The paper has the following limitations - Dedicated instruments are required to be mounted and utilized for the translation of the signs to speech. Portability is an issue as several devices including gloves and speakers need to be carried along with the computer for the database.

[8] Joyeeta Singha,Karen Das's research paper Recognition of Indian Sign Language in Live Video published in International Journal of Computer Applications (0975 – 8887) Volume 70– No.19 (2013) proposes a special purpose image processing algorithm based on Eigen vector to recognize various signs of Indian Sign Language for live video sequences with high accuracy. Skin Filtering is performed in the acquired video frames to get the Biggest Linked

Binary Object to detect the hand. The limitations of the paper are - extend the work from static image recognition of ISL to live video recognition and to improve accuracy of the model.

[9] Janice Light & David McNaughton's paper The Changing Face of Augmentative and Alternative Communication: Past, Present and Future Challenges, Augmentative and Alternative Communication (2012) gives information on AAC Systems. There are an increased number of AAC systems available, including unaided and aided systems that are not just dedicated speech generating devices, but also an increasing array of AAC software applications that are available on mainstream mobile technologies. The limitation of the paper is as follows - The paper does not share much technical insight into AAC. Much more work needs to be done to completely understand and document the effect it has on lives around the globe.

[10] Lisa A. Wood, Joanne Lasker, Ellin Siegel-Causey, David R. Beukelman, and Laura Ball's research paper Input Framework for Augmentative and Alternative Communication (1998) describes four components - augmenting the message, mapping language and symbols, augmenting retention, and developing a pool of response options. The paper has the following limitation - The AAC user may not comprehend the task itself, may have different associations to the symbol than the facilitator who chose it, or may not remember the appropriate symbol.

[11] Zangari, C., Lloyd, L., & Vicker, B' research paper Augmentative and alternative communication: An historic perspective. Augmentative and alternative communication, 10(1), 27-59 (1994) endeavors to record the social and noteworthy occasions that prompted the rise of the control of AAC and to recognize some significant achievements in its advancement. The paper recognizes the following limitation - It does not share much technical insight into AAC. It only provides a high level understanding of the concept.

Chapter 3

System Analysis and Design

3. System Analysis and Design

3.1 Overview

The Overview Section covers the methods used to design and develop the system. There are 2 major components in the system : Interface for Gesture Translation and Interface for Speech Translation (to text). System Analysis and Design consists of individual components and defining the Development Life Cycle (SDLC).

3.2 The System

The section covers the content specific to the system in detail starting from the requirements up to the collection of data.

3.2.1 System Characteristics

Data Collection of images is done initially using a phone camera. A dataset of images consisting of Indian Hand Signs is created. This is used to train the Convolution Neural Network (CNN) model.

The CNN algorithm is developed in Python language using Jupyter notebook as a platform for deployment and visualization. The CNN model was developed on a trial and error basis. Therefore a number of models were created with a combination of hyper-parameters and the best model was chosen.

The CNN Model used consists of 16 nodes, 3 Convolution Layers, 2 Dense Layers, Batch size of 64 and trained for 5 epochs.

3.2.2 System Assumptions, Constraints and Dependencies

Assumptions :

- The user of the system has knowledge of Indian Sign Language.
- Android platform to run and use the Application.

Constraints :

- Colors of Red and Brown interfere with the Detection of Skin if they are present in the focal region of the camera, leading to incorrect output.
- Lighting plays an important role in proper Detection and Classification with poor lighting leading to incorrect prediction.
- The hand signs of H, M and N are very similar and the model can sometimes give incorrect predictions as a result.

Dependencies :

- The device must be running Android 6.0 Marshmallow or higher for execution.
- Tensorflow dependencies on Android are required, integrated with the application.
- Camera and Recording (Speech and Microphone) permissions are required.

3.3 Requirements

Requirements are divided into Functional and User Interface Requirements.

3.3.1 Functional Requirements

- Algorithm (CNN) to train the model to identify and classify the character signs into 1 of 26 categories (A - Z).
- Requirement to view the result obtained after execution - words/sentence from Gestures.
- Requirement to view the result obtained after execution - words/sentence from Speech.

3.3.2 User Interface Requirements

- Two Modes of Operation of System.
- Real time display of the translated words to be displayed.
- Ease of use of Application
- Clean and Simple UI.

3.4 System Design

The System Design is as shown in the figure below :

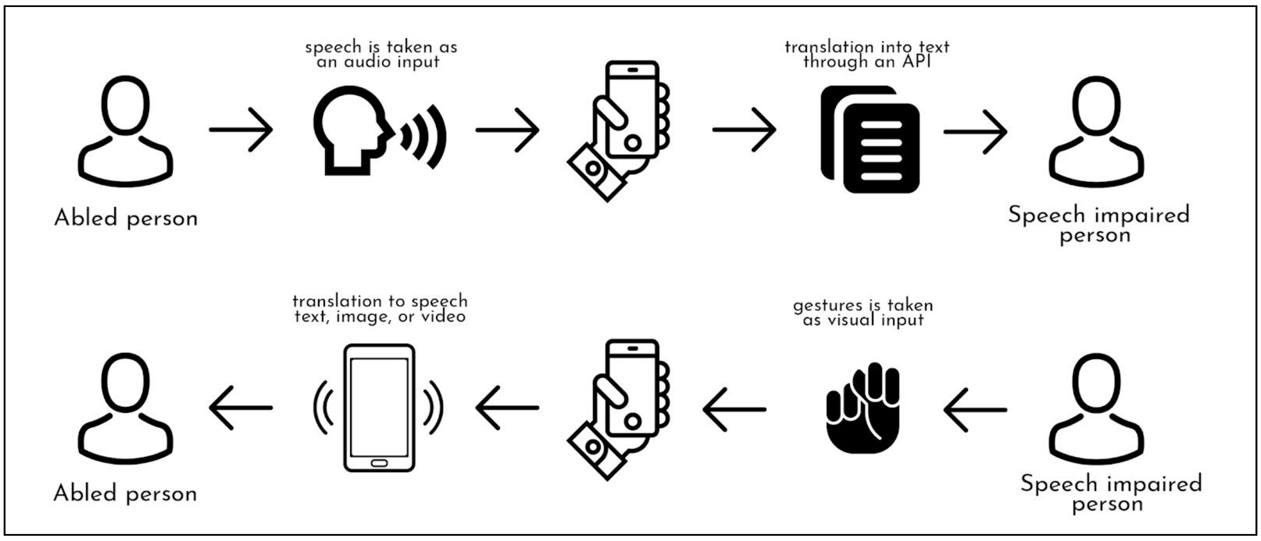

speech is taken as an audio input
translation into text through an API
Abled person
Speech impaired person
translation to speech text, image, or video
gestures is taken as visual input
Abled person
Speech impaired person

Fig 3.1 System Design

3.5 System Components

The following are the components of the proposed System

- Gesture -to- Text -to- Speech Translation Mode (using Machine Learning Algorithms and API's)
- Speech -to- Text Translation Mode (using API's)

3.5.1 Gesture Translation Mode

The Gesture Translation Mode is where the hand signs are mapped to their respective alphabets. The module offers Prediction by running the trained Convolution Neural Network (CNN) against the hand signs performed by the disabled person. The predicted alphabet is then displayed in a text view that has Text Completion capability so that the person may be able to select the most appropriate word suggestion that helps convey his/her message.

A Text-to-Speech API has also been incorporated into the Mode to allow the user of the Application to hear the sentence output by the Neural Network.

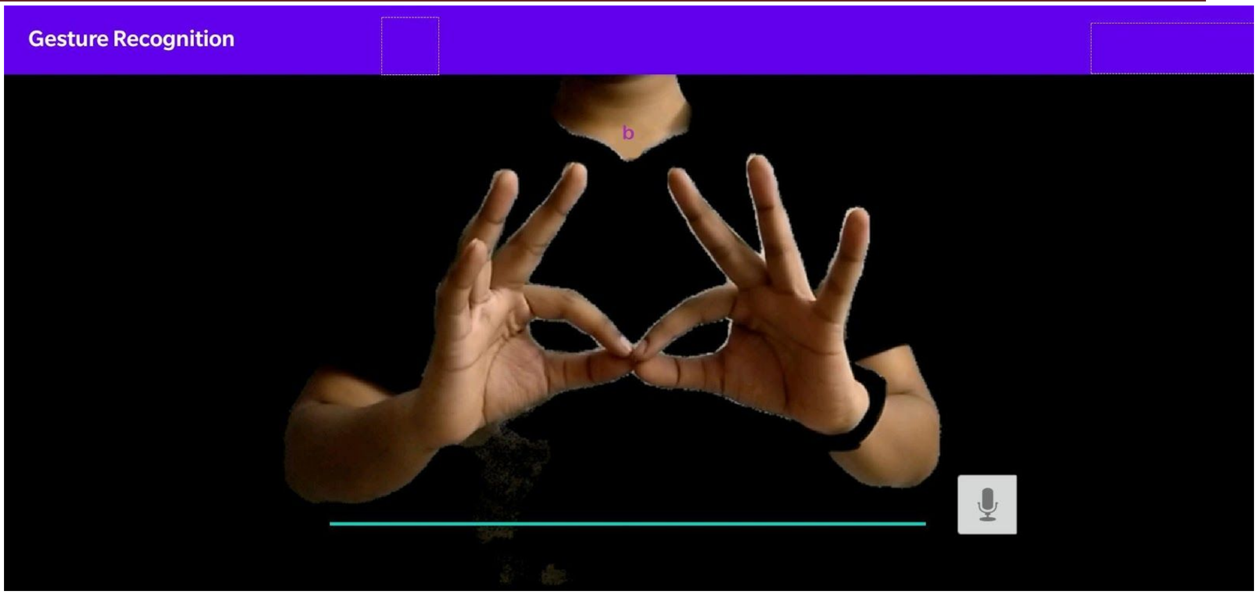


Fig 3.2 Gesture Translation Mode

3.5.2 Speech Translation Mode

The Speech Translation Mode is where the Speech Input from the Microphone is converted into Text. The converted Text is displayed in a Text View. The Mode makes use of Google's API which can be used online and offline. This mode allows users who are unable to hear to see the translated text by the User of the Application.

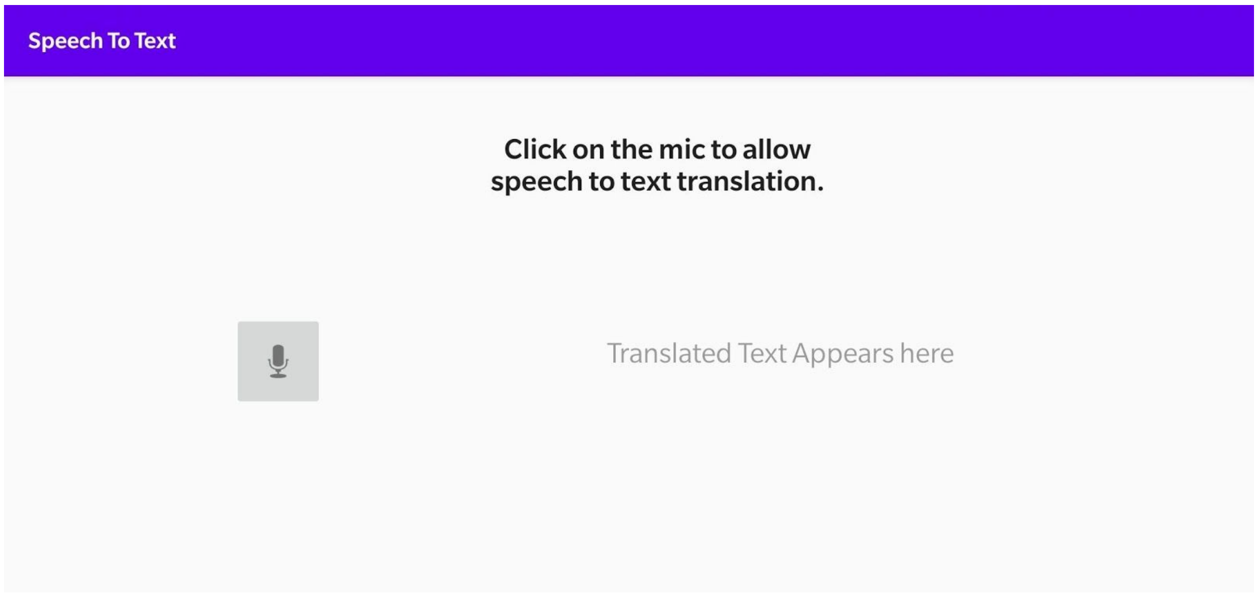

Speech To Text
Click on the mic to allow speech to text translation.
Translated Text Appears here

Fig 3.3 Speech Translation Mode

3.6 System Performance

The Validation Accuracy of the model before conversion to TensorFlow Lite model is 98%. The latency during Prediction is about 2 seconds.

3.7 Data Collection and Management

The dataset was created from scratch and comprises over 12,000 images of hand signs by various people. The data is only needed at time of training of the model and is saved on Google Drive (one time storage, does not require repeated access). Google Drive was used as it provides free, large storage space and ease of use when coupled with Google Colab (which was initially used for preprocessing).

Chapter 4
Modeling and Implementation

4. Modeling and Implementation

4.1 Overview

This chapter covers the following subjects

- The Languages, Tools used to Model the System
- Flow Diagram
- Sequence Diagram
- Implementation of Modules

4.2 Languages and Tools Used

The following are the various Tools and Technologies used to model and develop the system consisting of front-end and back-end services.

- Data Analysis
    1. Language - Python (3.7)
    2. IDE - Jupyter Notebook
- Python Modules
    1. numpy
    2. tensorflow
    3. keras
    4. pickle
    5. opencv
- Data Storage
    1. Image dataset hosted on Google Drive
- Mobile Application
    1. Language - JAVA
    2. IDE - Android Studio

4.3 Flow Diagram

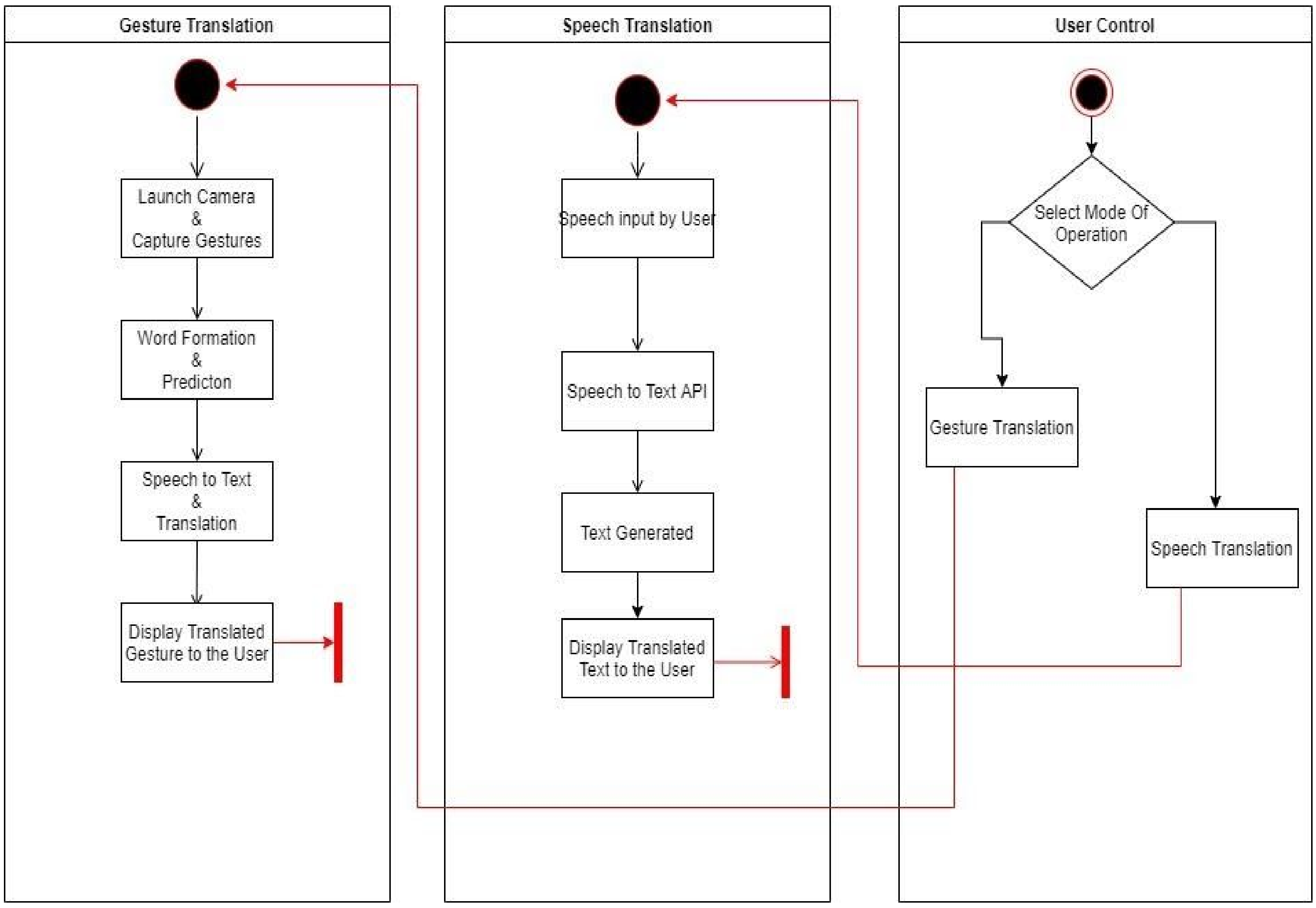


Fig 4.1 System Flow Diagram

The User Control allows switching between the 2 Modes of Operation - Gesture Translation and Speech Translation.

The Gesture Translation Mode translates Gestures to words and then to Speech. The Mode contains 4 modules -

1) **Launch Camera and Capture the Gesture** performed by the User to process the frames.
2) **Word Formation and Prediction** to give a character prediction based on the trained images.
3) **Display Translated Gesture** to the User as a word/sentence.
4) **Text to Speech Translation** to convert the generated words to Speech output.

The Speech Translation Mode translates Speech to Text. The Mode contains 4 modules -

1) **Speech Input by the User** is recorded by the Microphone.
2) **The Speech-to-Text API** is called to process the Audio input.
3) **The Text is generated** after Translation.
4) **Display the Translated Text to the User** in a TextView.

4.4 Sequence Diagram

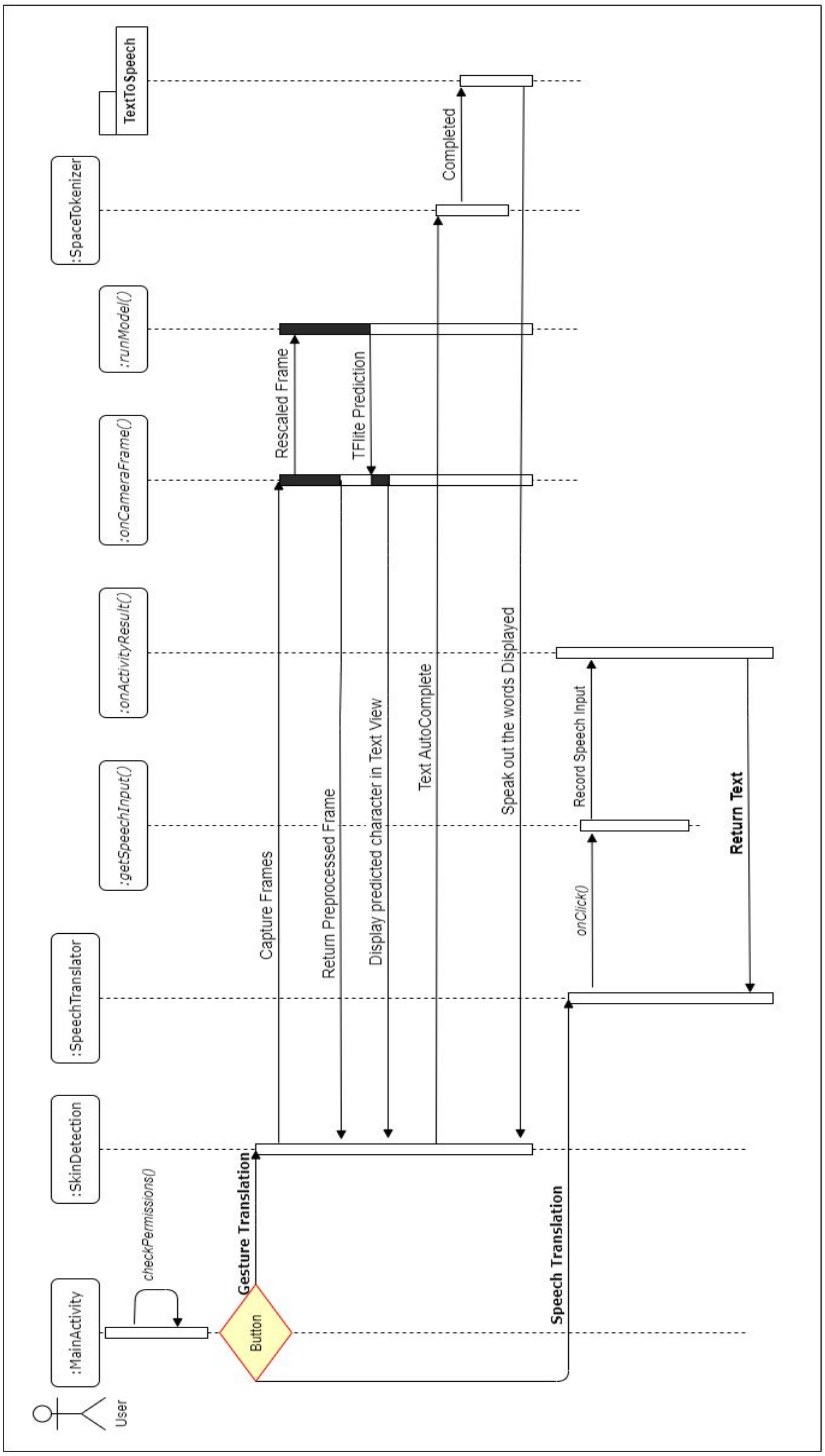


Fig 4.2 System Sequence Diagram

The Application is started by the user at the MainActivity. First task performed is a check for Camera Permissions (necessary in order to use the application). From there on the Activity has 2 Buttons leading to SkinDetection.class or SpeechTranslation.class based on User choice.

SkinDetection contains a function onCameraFrame() which processes the frames received from the camera. The image preprocessing and filtering for skin is done here and the processed frame is returned to be displayed on screen. The function passes the frame to runModel() where the frame is scaled and passed to the tflite model for prediction. The model.tflite in the assets folder containing the weights and values is run against the frame.

The prediction (A or B or C) is passed to a textbox that is displayed on the Activity. From here on the textbox contains the AutoComplete Android API and starts providing suggestions for words to the user. This process is repeated as long as the user is signing out characters.

Once the process is complete SpaceTokenizer is called to add appropriate annotations and it further calls a Text-to-Speech API that voices out the converted word(s) for accessibility to the user.

SpeechTranslator contains the Google Speech-to-Text API to allow the conversion of speech to words that are displayed on the screen. The Activity contains a button, when clicked calles the method onClick() and starts recording the audio. The function checks if the phone can record audio and then the processed result is passed from onActivityResult() to the User to be displayed in a TextView.

4.5 Implementation

Steps:

1. Dataset Collection
2. Analysis & Result

4.5.1 Dataset Collection

There was no pre-existing dataset for Indian Sign Language, one had to be created. 26 ISL gestures representing 26 letters of the alphabet were considered for making the dataset. Approximately 500 images for each gesture were taken on a smartphone involving 11 individuals, making a dataset of approximately 13000 images. The pictures had a resolution of 1920x1080 and were taken in landscape mode.

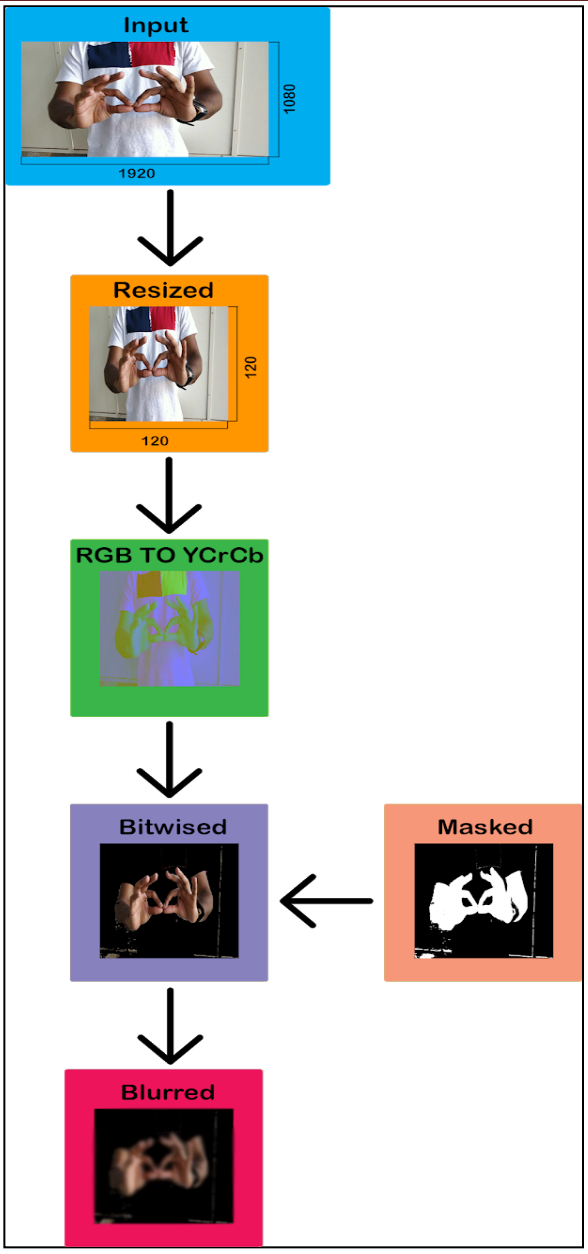
4.5.2 Analysis & Result

4.5.2.1 Image Preprocessing

Preprocessing is done in order to retain the necessary elements of the picture and discard the redundant elements of the same. The necessary elements comprise the skin and the gestures and redundant elements comprise the background. In order to segregate the gestures from the background we used the following preprocessing steps.

- The picture is first resized to a resolution of 120x120 from 1920x1080 without losing any critical features in order to reduce the size of the input data for the CNN model hence making it easier to process.
- The image is then transformed from the RGB scale to YCrCb scale to facilitate skin detection in the image.
- A specific YCrCb (Y, green, is the luma component and C_B and C_R are the blue-difference and red-difference chroma components) range is defined which matches the range of values that approximately defines the skin color.
- This range is applied on the resized image to obtain a mask where skin/gesture is identified as white pixels and the rest of the picture is identified as black pixels.
- When 'bitwise_and' operation is performed on the resized image using the generated mask, an image containing only the skin/gesture is obtained which is used to train the model.

All the preprocessing steps were performed using the OpenCV library available for python.


Input
1080
1920
Resized
120
120
RGB TO YCrCb
Bitwised
Masked
Blurred

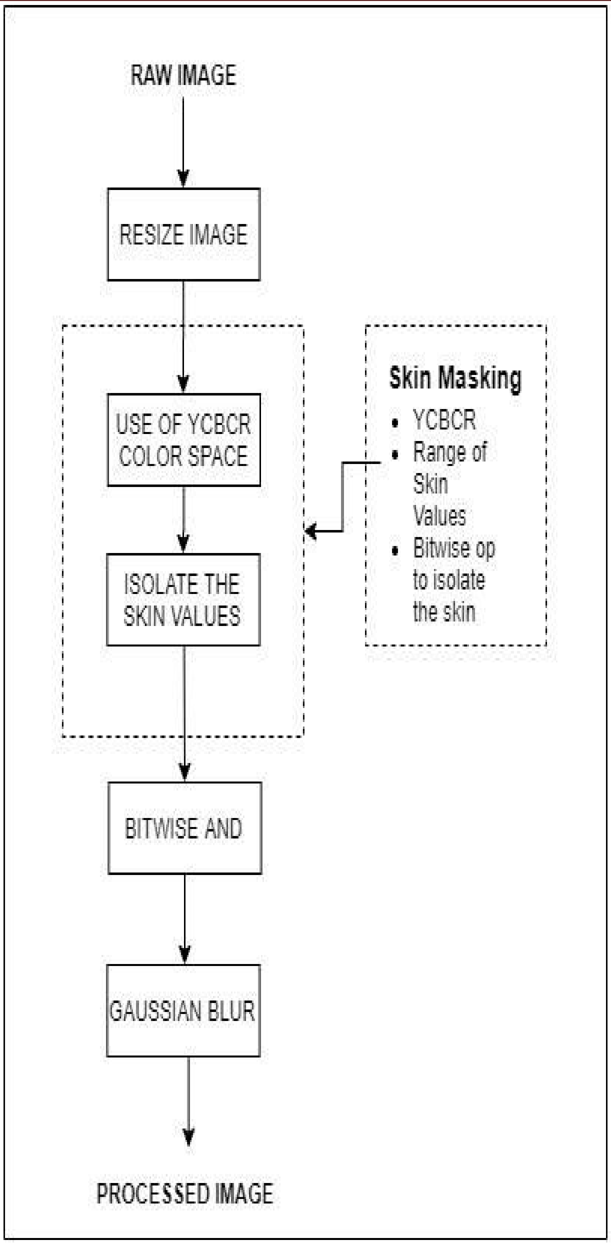
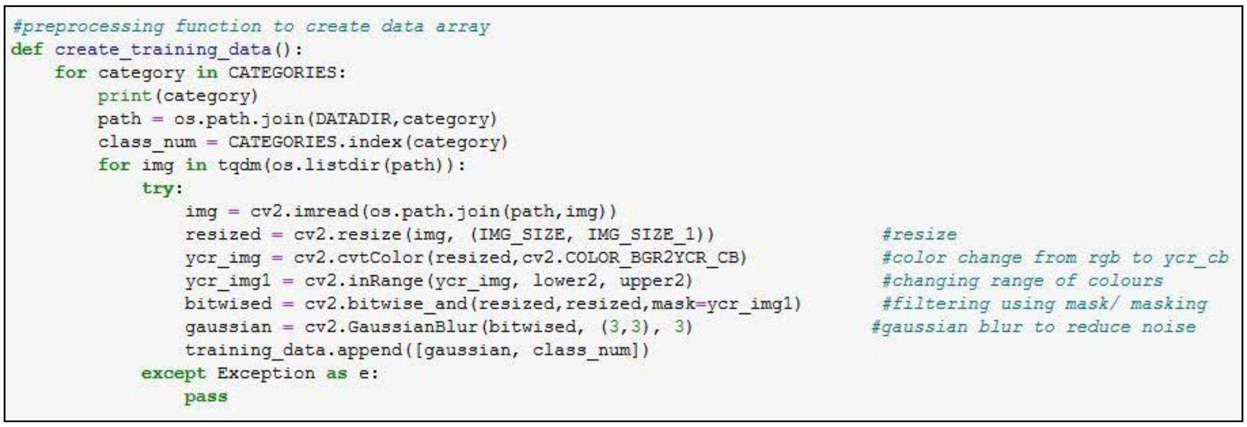


Fig 4.3 Image Preprocessing Stages



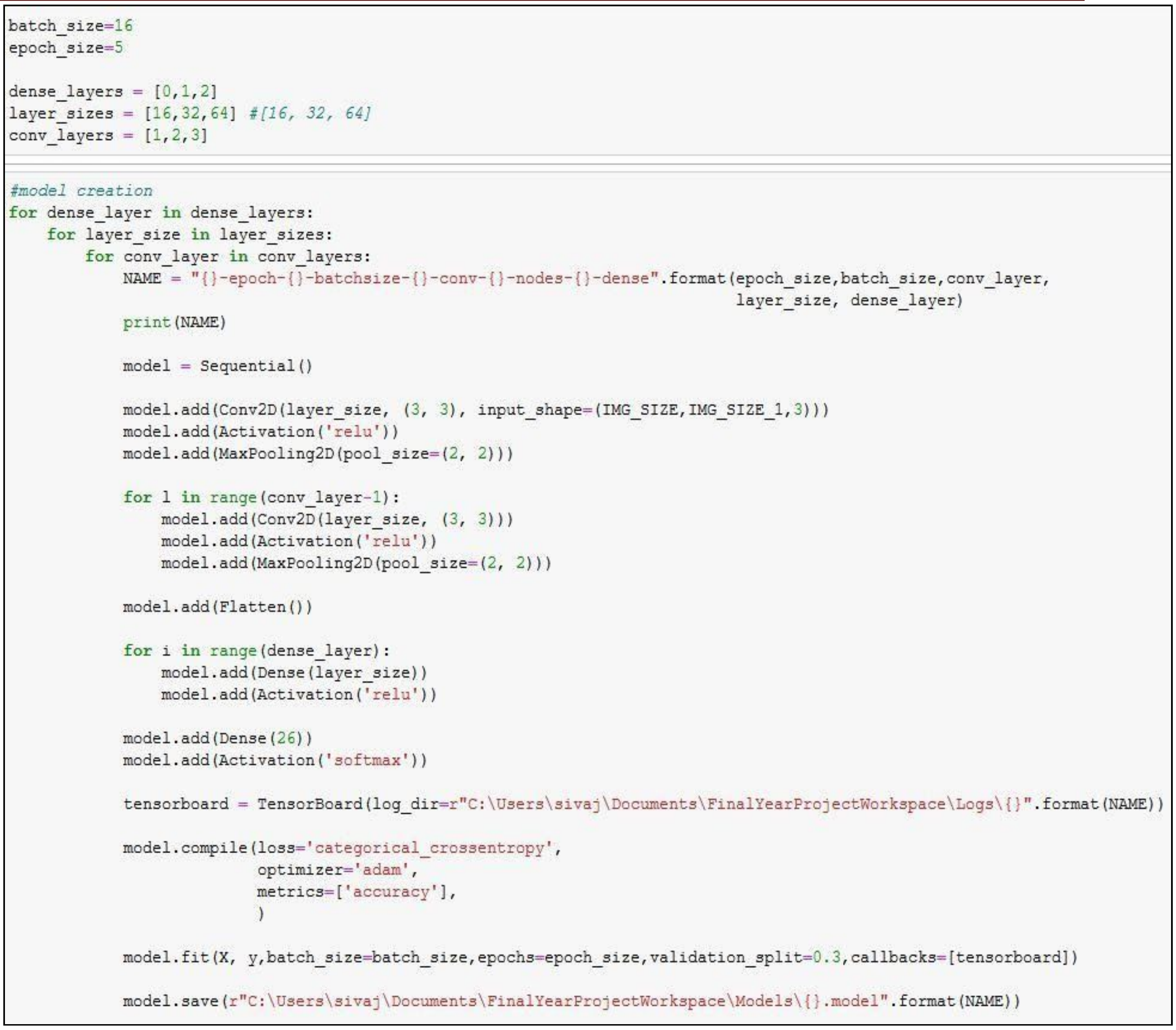
```
#preprocessing function to create data array
def create_training_data():
    for category in CATEGORIES:
        print(category)
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)
        for img in tqdm(os.listdir(path)):
            try:
                img = cv2.imread(os.path.join(path,img))
                resized = cv2.resize(img, (IMG_SIZE, IMG_SIZE_1))             #resize
                ycr_img = cv2.cvtColor(resized,cv2.COLOR_BGR2YCR_CB)          #color change from rgb to ycr_cb
                ycr_img1 = cv2.inRange(ycr_img, lower2, upper2)               #changing range of colours
                bitwised = cv2.bitwise_and(resized,resized,mask=ycr_img1)     #filtering using mask/ masking
                gaussian = cv2.GaussianBlur(bitwised, (3,3), 3)              #gaussian blur to reduce noise
                training_data.append([gaussian, class_num])
            except Exception as e:
                pass
```

Fig 4.4 Image Preprocessing Code Snippet

4.5.2.2 Model Generation

The chosen model was generated along with many others on a trial and error basis, generating numerous models based on a combination of the following parameters - layer size, epochs, dense layers and convolution layers. This led to the creation of 81 models (3^4). Out of these models the most appropriate one was chosen which had high validation accuracy and minimal loss and did not over fit the data.

The architecture of the CNN consists of Convolution Layers, Dense Layers, Activation Functions and Max Pooling Functions. An 'Adam' Optimizer is used which stands for Adaptive Learning which is used to find individual learning rates for each parameter and helps in training the model. The model is finally saved and the logs for each model can be accessed via tensorboard.



```
batch_size=16
epoch_size=5

dense_layers = [0,1,2]
layer_sizes = [16,32,64] #[16, 32, 64]
conv_layers = [1,2,3]
```

```
#model creation
for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
            NAME = "{}-epoch-{}-batchsize-{}-conv-{}-nodes-{}-dense".format(epoch_size,batch_size,conv_layer,
                                                                            layer_size, dense_layer)
            print(NAME)

            model = Sequential()

            model.add(Conv2D(layer_size, (3, 3), input_shape=(IMG_SIZE,IMG_SIZE_1,3)))
            model.add(Activation('relu'))
            model.add(MaxPooling2D(pool_size=(2, 2)))

            for l in range(conv_layer-1):
                model.add(Conv2D(layer_size, (3, 3)))
                model.add(Activation('relu'))
                model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Flatten())

            for i in range(dense_layer):
                model.add(Dense(layer_size))
                model.add(Activation('relu'))

            model.add(Dense(26))
            model.add(Activation('softmax'))

            tensorboard = TensorBoard(log_dir=r"C:\Users\sivaj\Documents\FinalYearProjectWorkspace\Logs\{}".format(NAME))

            model.compile(loss='categorical_crossentropy',
                          optimizer='adam',
                          metrics=['accuracy'],
                          )

            model.fit(X, y,batch_size=batch_size,epochs=epoch_size,validation_split=0.3,callbacks=[tensorboard])

            model.save(r"C:\Users\sivaj\Documents\FinalYearProjectWorkspace\Models\{}.model".format(NAME))
```

Fig 4.5 CNN Models Generation Code Snippet

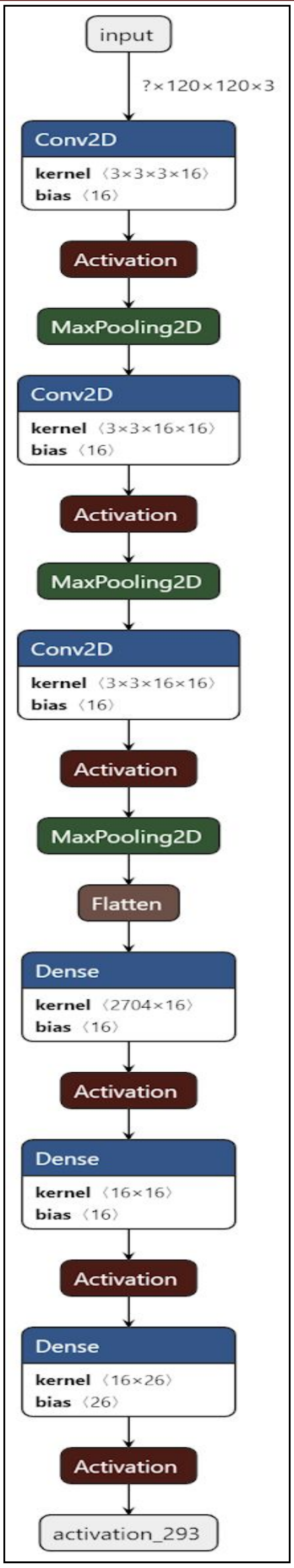


Fig 4.6 Layers of the Chosen CNN Model

Fig <fig.no> shown above depicts a flow diagram of the various layers of the CNN model used. The architecture in a CNN is analogous to that of the connectivity pattern on the Brain.

The input into the model is an image (or a frame, in this case) which is composed of pixels. As the images are 120x120 the input size to the first Layer is 120x120x3 (width X height X number of channels).

The First Layer is a Convolution Layer. The input image is convolved using a Kernel/Filter. We have used a 3x3x3x16 matrix for the same. The filter is moved across the image pixel by pixel and the new pixel values are formed (convolved). Convolution is done in order to extract high level features like edges from the input image. The model has a total of 3 Convolution Layers.

Pooling Layer reduces the spatial size of the Convolved Feature. This limits the power required to process data. This is known as dimensionality reduction. It also performs de-noising and extracts the dominant features in an image. The Pooling method applied is Max Pooling - which returns the maximum value from the portion of the image imposed by the kernel.

Dense Layers are also known as Fully Connected Layers. It is used to learn non linear combinations output by the convolution layer.

Activation Function used is ‘ReLU’ which stands for ‘Rectified Linear Unit’ and is defined as y = max(0,x), where x is the input to a neuron. ReLU can be visualized as a ramp function defined in the positive part of its argument.

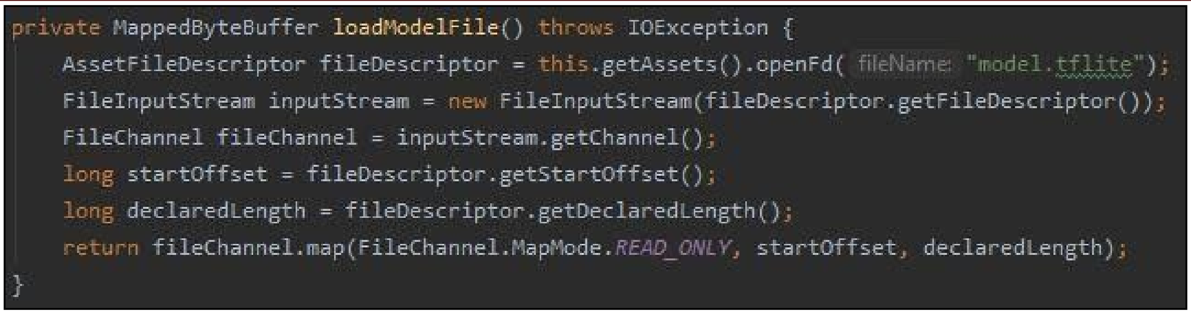
The image needs to be Flattened to a column vector which is fed to the NN and backpropagation is applied in each epoch.

Another activation function used is ‘Softmax’ which turns the numbers into probabilities (which add up to 1). The output is a vector which represents the probability distributions of a list of potential outcomes.

4.5.2.3 Moving model to Android and result

Android devices do not support running of the generated Keras H5 CNN model and hence it is converted to a TensorFlow Lite CNN model using the TensorFlow Model Converter. As mentioned previously, two modes of operation are provided by the Android Application and each following a specific sequence of events.

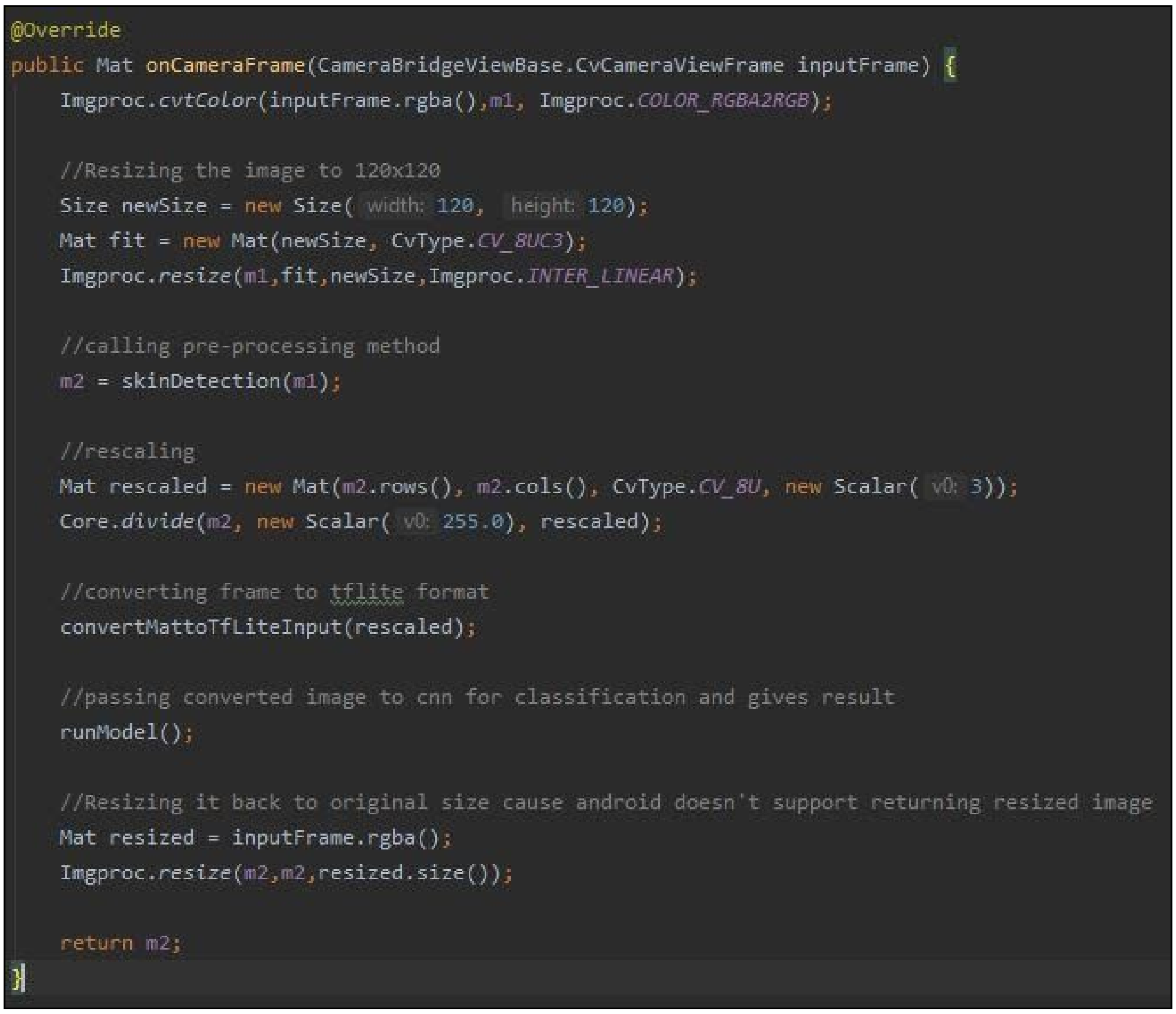
For the Gesture -to- Text -to- Speech mode, the TensorFlow Lite model is first loaded in the memory using the ‘loadModelFile()’ function, the model comes with the apk file. Once the model is loaded, on every frame captured by the smartphone the ‘onCameraFrame()’ method is executed.



```java
private MappedByteBuffer loadModelFile() throws IOException {
    AssetFileDescriptor fileDescriptor = this.getAssets().openFd( fileName: "model.tflite");
    FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
    FileChannel fileChannel = inputStream.getChannel();
    long startOffset = fileDescriptor.getStartOffset();
    long declaredLength = fileDescriptor.getDeclaredLength();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}
```

Fig 4.7 loadModelFile Method Code Snippet

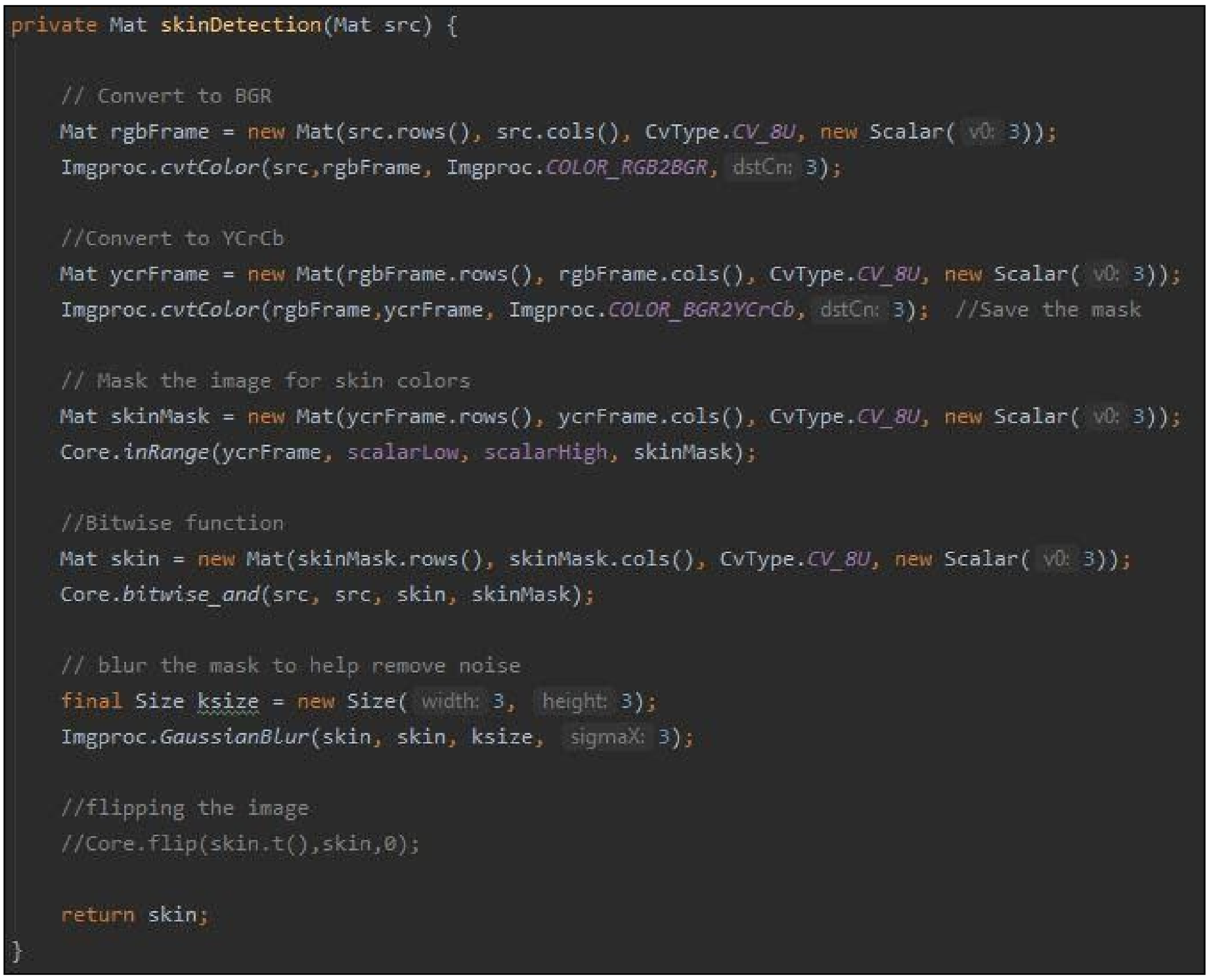
Every frame captured by an Android smartphone is of the RGBA format and needs to be converted to the RGB format inorder to apply the same preprocessing steps that were applied for the training images; the OpenCV library available for Android is used for the same. The RGB frame is passed to the 'skinDetection()' function to apply preprocessing.



```java
@Override
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    Imgproc.cvtColor(inputFrame.rgba(),m1, Imgproc.COLOR_RGBA2RGB);

    //Resizing the image to 120x120
    Size newSize = new Size( width: 120,  height: 120);
    Mat fit = new Mat(newSize, CvType.CV_8UC3);
    Imgproc.resize(m1,fit,newSize,Imgproc.INTER_LINEAR);

    //calling pre-processing method
    m2 = skinDetection(m1);

    //rescaling
    Mat rescaled = new Mat(m2.rows(), m2.cols(), CvType.CV_8U, new Scalar( v0: 3));
    Core.divide(m2, new Scalar( v0: 255.0), rescaled);

    //converting frame to tflite format
    convertMattoTfLiteInput(rescaled);

    //passing converted image to cnn for classification and gives result
    runModel();

    //Resizing it back to original size cause android doesn't support returning resized image
    Mat resized = inputFrame.rgba();
    Imgproc.resize(m2,m2,resized.size());

    return m2;
}
```

Fig 4.8 onCameraFrame Method Code Snippet

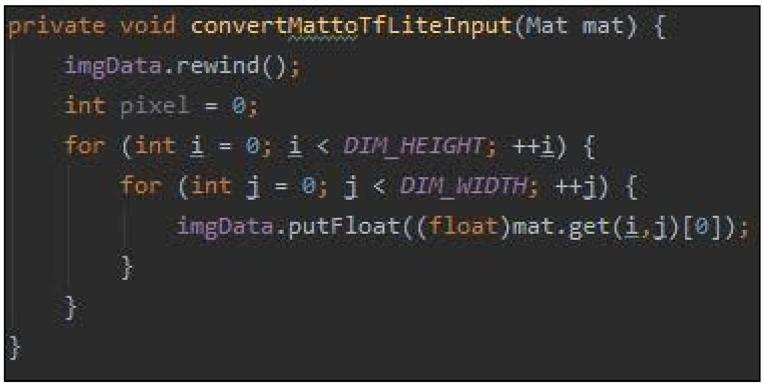
The function 'skinDetection()' replicates the steps performed in the python preprocessing. The functions available in the android OpenCV library are used for the same. The method returns a preprocessed image (frame) to the calling function.



```java
private Mat skinDetection(Mat src) {

    // Convert to BGR
    Mat rgbFrame = new Mat(src.rows(), src.cols(), CvType.CV_8U, new Scalar( v0: 3));
    Imgproc.cvtColor(src,rgbFrame, Imgproc.COLOR_RGB2BGR, dstCn: 3);

    //Convert to YCrCb
    Mat ycrFrame = new Mat(rgbFrame.rows(), rgbFrame.cols(), CvType.CV_8U, new Scalar( v0: 3));
    Imgproc.cvtColor(rgbFrame,ycrFrame, Imgproc.COLOR_BGR2YCrCb, dstCn: 3);  //Save the mask

    // Mask the image for skin colors
    Mat skinMask = new Mat(ycrFrame.rows(), ycrFrame.cols(), CvType.CV_8U, new Scalar( v0: 3));
    Core.inRange(ycrFrame, scalarLow, scalarHigh, skinMask);

    //Bitwise function
    Mat skin = new Mat(skinMask.rows(), skinMask.cols(), CvType.CV_8U, new Scalar( v0: 3));
    Core.bitwise_and(src, src, skin, skinMask);

    // blur the mask to help remove noise
    final Size ksize = new Size( width: 3,  height: 3);
    Imgproc.GaussianBlur(skin, skin, ksize,  sigmaX: 3);

    //flipping the image
    //Core.flip(skin.t(),skin,0);

    return skin;
}
```

Fig 4.9 skinDetection Method Code Snippet

The function convertMatToTfLiteInput() converts the preprocessed frame into a format that uses the TensorFlow Lite model as input for prediction and saves it in the imgData variable.

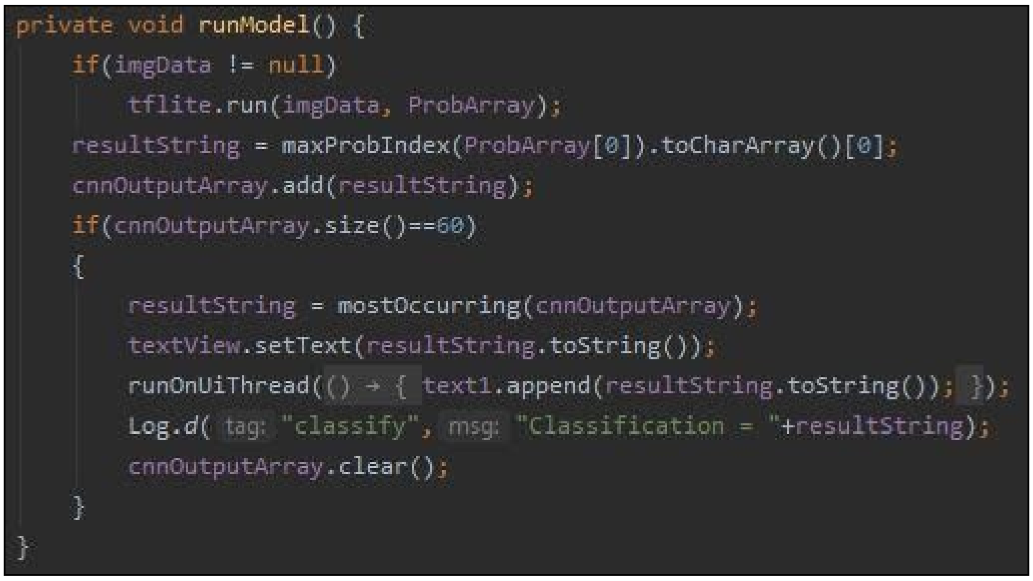


```java
private void convertMattoTfLiteInput(Mat mat) {
    imgData.rewind();
    int pixel = 0;
    for (int i = 0; i < DIM_HEIGHT; ++i) {
        for (int j = 0; j < DIM_WIDTH; ++j) {
            imgData.putFloat((float)mat.get(i,j)[0]);
        }
    }
}
```

Fig 4.10 convertMattoTfLiteInput Method Code Snippet

The function ‘runModel()’ runs the tflite model and predicts the alphabet the gesture represents, this is appended to the cnnOutputArray. Once every 60 predictions, the most frequently occurring character from the cnnOutputArray is displayed to the user.

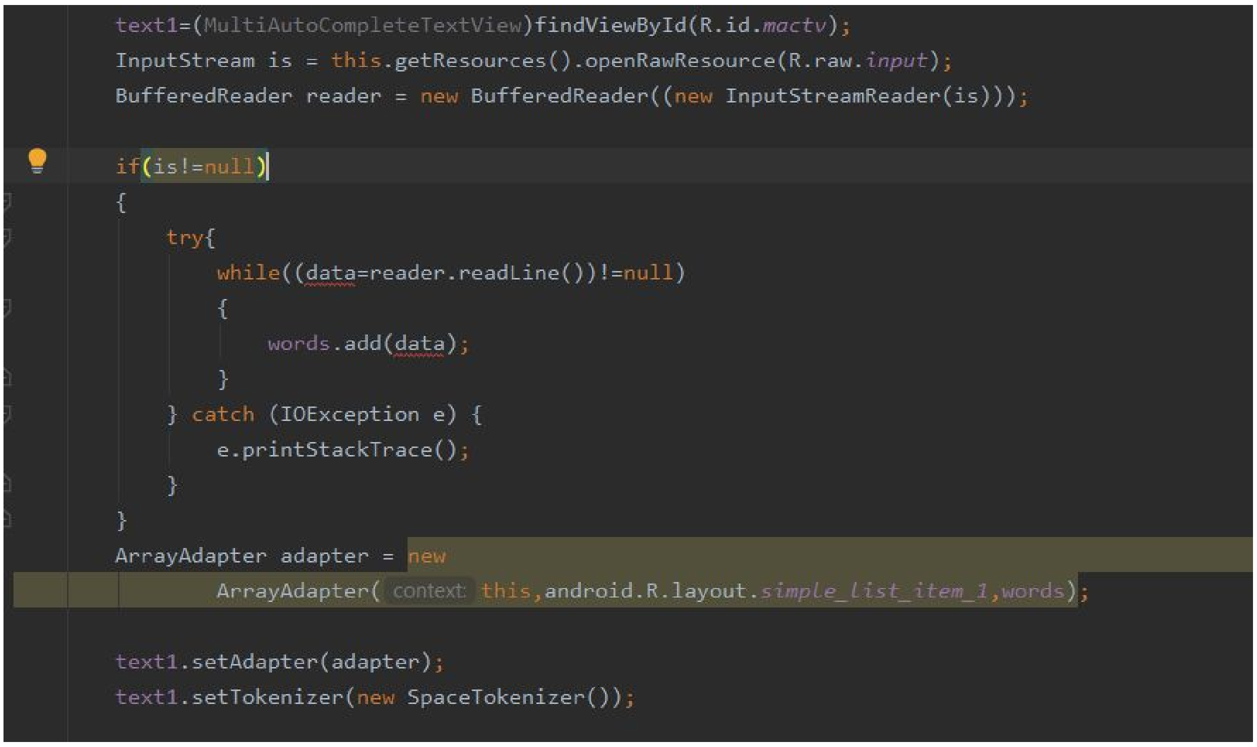
There are 24 frames captured each second, and if 24 predictions are considered without any averaging, there will be 24 characters predicted for each second which isn’t viable. Hence, it was decided to take the most occuring prediction for 60 frames of input.



```java
private void runModel() {
    if(imgData != null)
        tflite.run(imgData, ProbArray);
    resultString = maxProbIndex(ProbArray[0]).toCharArray()[0];
    cnnOutputArray.add(resultString);
    if(cnnOutputArray.size()==60)
    {
        resultString = mostOccurring(cnnOutputArray);
        textView.setText(resultString.toString());
        runOnUiThread(() → { text1.append(resultString.toString()); });
        Log.d( tag: "classify", msg: "Classification = "+resultString);
        cnnOutputArray.clear();
    }
}
```

Fig 4.11 runModel Method Code Snippet

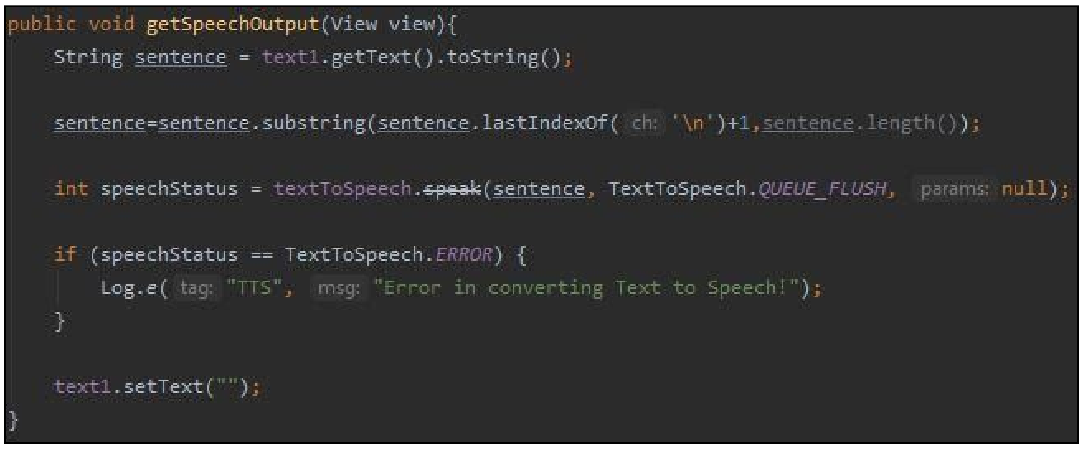
The autocompletion of words was accomplished using androids MultiAutoCompleteTextView. The words dataset was populated into the MultiAutoCompleteTextView. A SpaceTokenizer class was created to distinguish different substrings in the textbox. The autocompleter works by taking the string in between the last space token found until the last character typed and triggers a dropdown with the suggested words.



```
text1=(MultiAutoCompleteTextView)findViewById(R.id.mactv);
InputStream is = this.getResources().openRawResource(R.raw.input);
BufferedReader reader = new BufferedReader((new InputStreamReader(is)));

if(is!=null)
{
    try{
        while((data=reader.readLine())!=null)
        {
            words.add(data);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
ArrayAdapter adapter = new
        ArrayAdapter( context: this,android.R.layout.simple_list_item_1,words);

text1.setAdapter(adapter);
text1.setTokenizer(new SpaceTokenizer());
```

Fig 4.12 Adding words to MultiAutoCompleteTextView and setting adapter and tokenizer

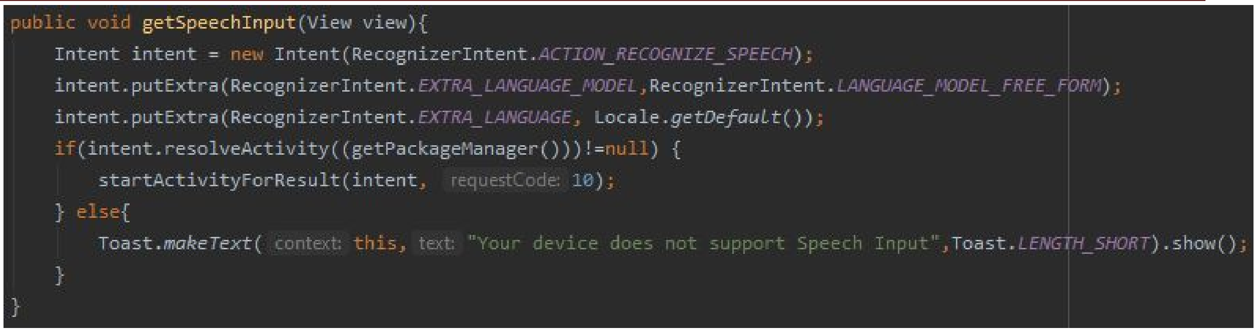
Once the words are generated as needed. The user can use the Speech Button to voice out the predicted text. The method getSpeechOutput() performs this task. The text is first read into a String variable and then the speak() function is called on it. A QUEUE_FLUSH is performed to prevent the API from repeating previously generated sentences.



```
public void getSpeechOutput(View view){
    String sentence = text1.getText().toString();

    sentence=sentence.substring(sentence.lastIndexOf( ch: '\n')+1,sentence.length());

    int speechStatus = textToSpeech.~~speak~~(sentence, TextToSpeech.QUEUE_FLUSH,  params: null);

    if (speechStatus == TextToSpeech.ERROR) {
        Log.e( tag: "TTS",  msg: "Error in converting Text to Speech!");
    }

    text1.setText("");
}
```

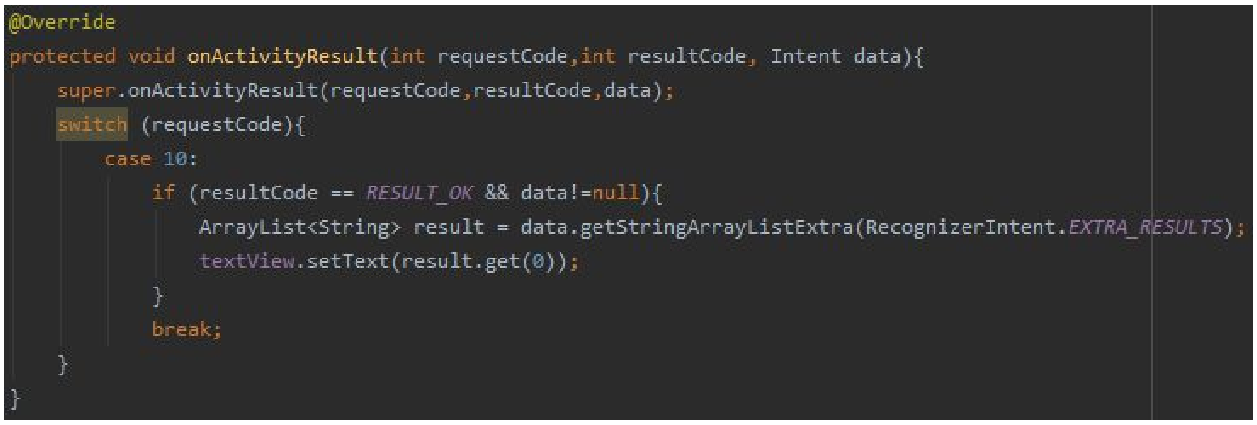
Fig 4.13 getSpeechOutput Method Code Snippet

For the Speech -to- Text mode, an API is used to record a voice note at the click of a button which is then converted to text and displayed to the user using a text box.



```java
public void getSpeechInput(View view){
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    if(intent.resolveActivity((getPackageManager()))!=null) {
        startActivityForResult(intent,  requestCode: 10);
    } else{
        Toast.makeText( context: this,  text: "Your device does not support Speech Input",Toast.LENGTH_SHORT).show();
    }
}
```

Fig 4.14 getSpeechInput Method Code Snippet



```java
@Override
protected void onActivityResult(int requestCode,int resultCode, Intent data){
    super.onActivityResult(requestCode,resultCode,data);
    switch (requestCode){
        case 10:
            if (resultCode == RESULT_OK && data!=null){
                ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                textView.setText(result.get(0));
            }
            break;
    }
}
```

Fig 4.15 onActivityResult Method Code Snippet

Chapter 5

Testing, Results and Discussion

5. Testing, Results and Discussion

5.1 Testing

5.1.1 Image Preprocessing

Three preprocessing methodologies were tested to determine one which could isolate the region of interest (hand gesture/ skin) with the best quality.

The first method converted the original image into a black and white image with just one dimension. The region of interest was then extracted using the ‘bitwise and’ feature of OpenCV. Unfortunately, the background noise was being considered as the region of interest with this method. Unnecessary background noise would affect the accuracy of the predictions in the future and hence was discarded.

The second method involved applying a mask to the original RGB image, the mask was created using the skin color range on the RGB scale. Once the mask was applied, the region of interest was extracted with a better accuracy as compared to the first method. Gaussian Blur was used to reduce the noise. When models were trained using images that were preprocessed using this method, they were not as accurate as the models that were trained using a dataset prepared using the following method.

The third method is what was used, the same has been explained in Chapter 4.

5.1.2 Speech-to-Text Translation

The Speech to Text Translation is accomplished on Android by making use of Google’s API which comes pre-installed in all Android smartphones. The API can work with or without the use of the internet (offline use only requires the download of the language package). As it is an API, multiple language options are also available. It does best to reduce noise while capturing audio but for most accurate results it is preferred that the user must speak clearly into the microphone.

5.1.3 Machine Learning Algorithm

The Algorithm used to train and classify the images is a Convolution Neural Network (CNN). A CNN was chosen as it is specially designed to deal with image data and has specific functions for the same. It makes processing easier by assigning weights to more important features in the image. Other classifiers such as SVM, Random Forests may also be used but require adequate feature engineering without which the accuracy of the model decreases.

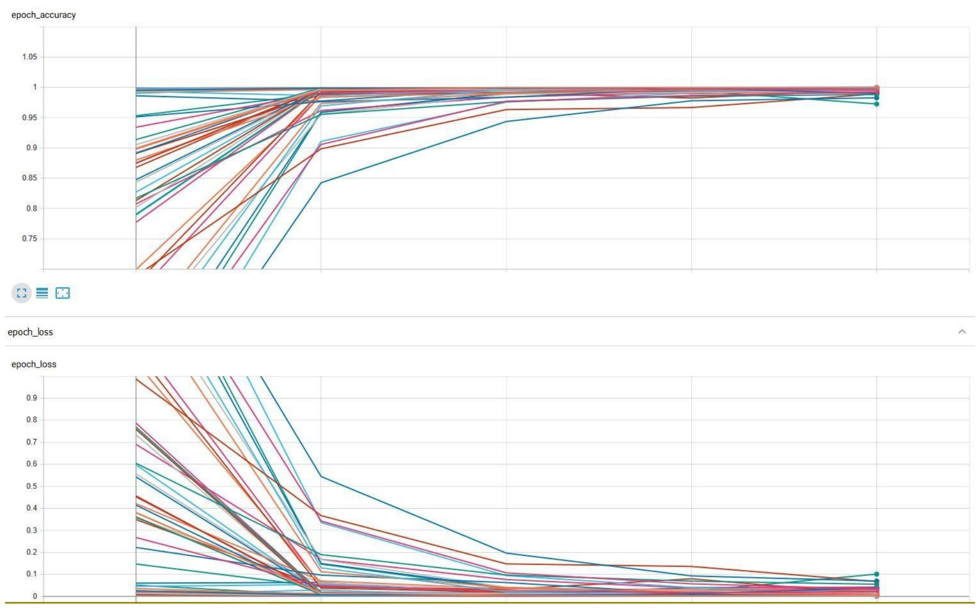


Fig 5.1 Tensorboard Graphs of all CNN Models

Numerous models were trained and tested to determine the most consistent and accurate model. Models were different with respect to the number of convolution layers used, number of dense layers used, number of epochs used, the batch size of the input data, number of nodes present at each layer. All models were compared using tensorboard's accuracy and error graphs.

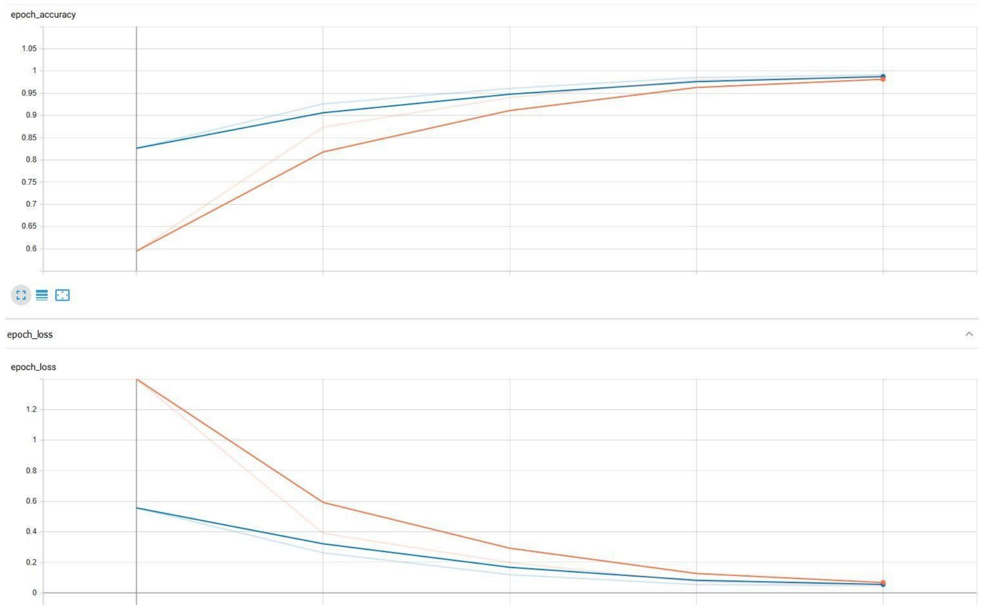


Fig 5.2 Tensorboard Graph of Chosen CNN model

5.1.4 Android Application

The application has a camera requirement, for which permissions are required from the User. The application was run on numerous Android devices and it ran successfully only on systems with Android 6.0 or above. It failed to execute completely on some compatible devices possibly due to processor constraints.

5.2. Results

The end result is an Android application that opens a landscape homepage shown below. It has two buttons that represent the two modes of operation as discussed previously.

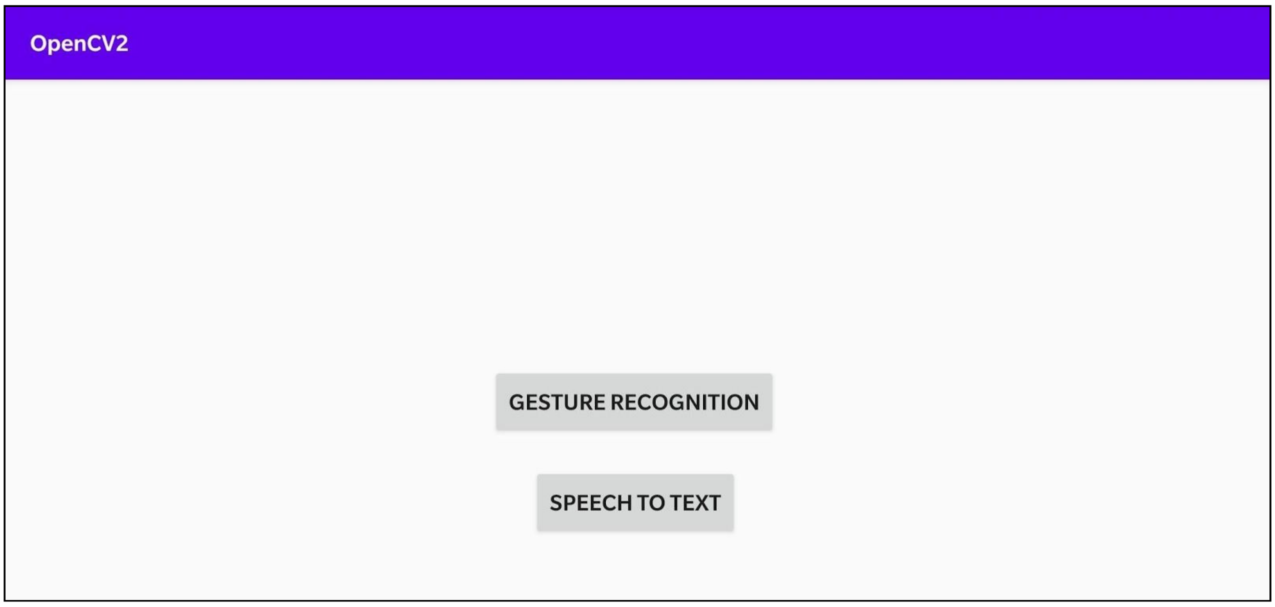


Fig 5.3 Android Application Home Page

On clicking Gesture Recognition the user will be taken to a new page which is shown below. In this mode the smartphone's rear camera will be used to capture frames of the gestures for prediction. The individual predictions will be displayed in a text box on the top of the screen. The word predictions will be displayed in the text box beside the microphone button. On clicking the microphone button, the predicted sentences will be outputted through the smartphone's speaker.

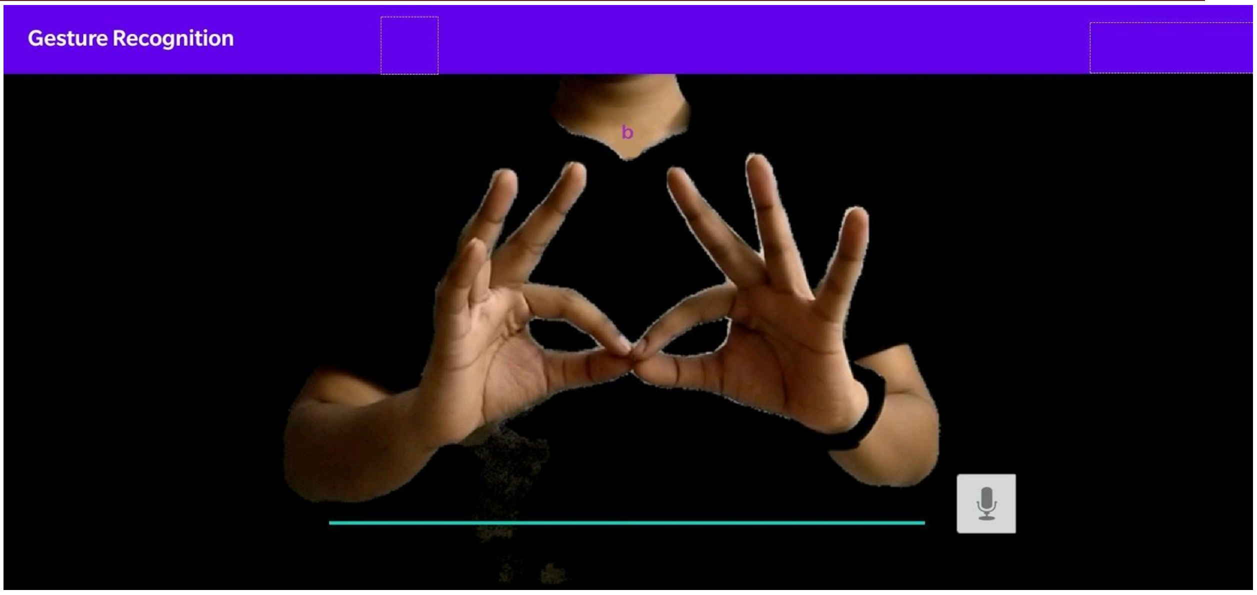


Fig 5.4 Android Application Gesture Translation Mode

When the Speech To Text button is clicked on the home page, the user will be taken to a page that is shown below. In this mode the user can record speech, once that is converted to text it will be displayed in the text box.

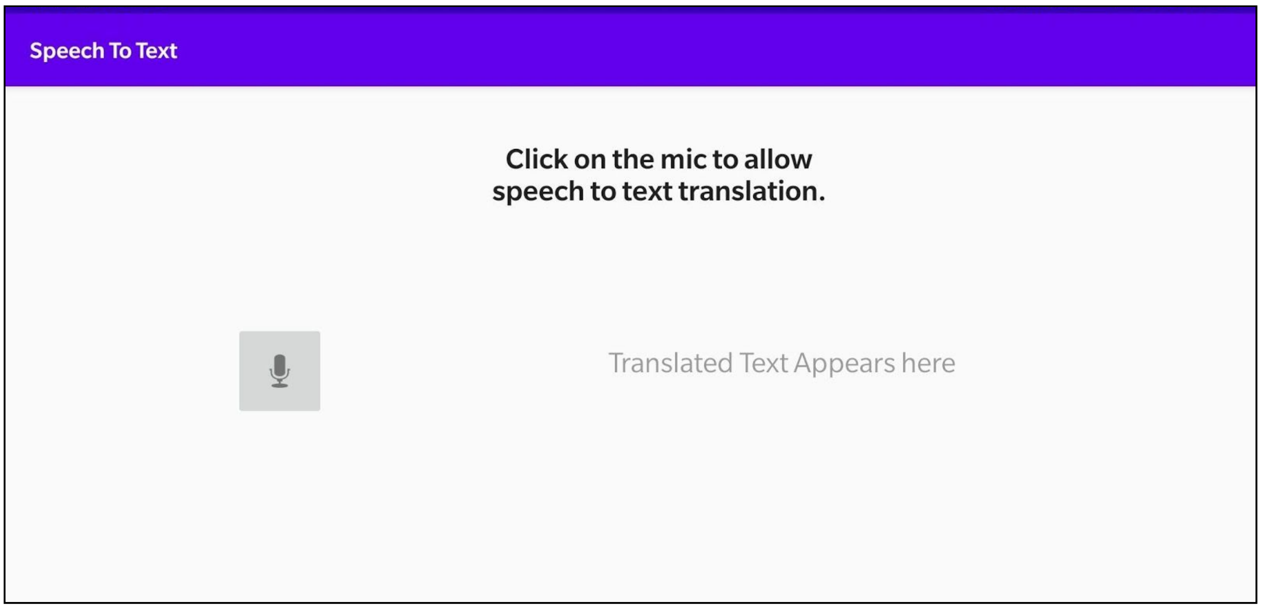

Speech To Text
Click on the mic to allow speech to text translation.
Translated Text Appears here

Fig 5.5 Android Application Speech Translation Mode

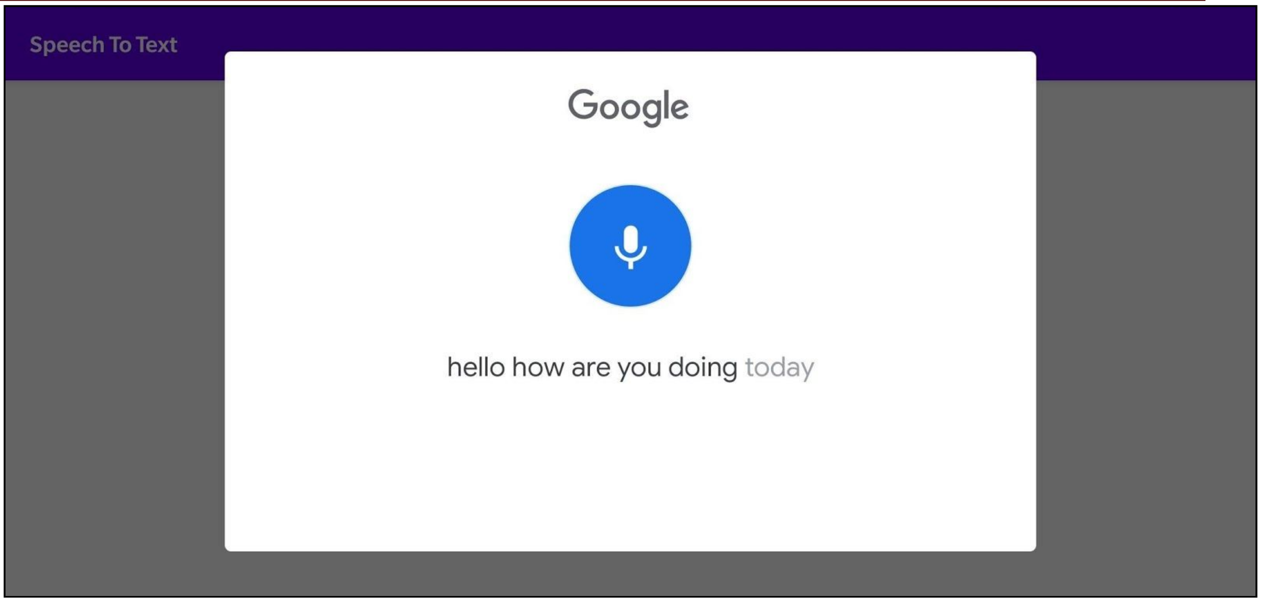


Fig 5.6 Google API to Record Speech Input

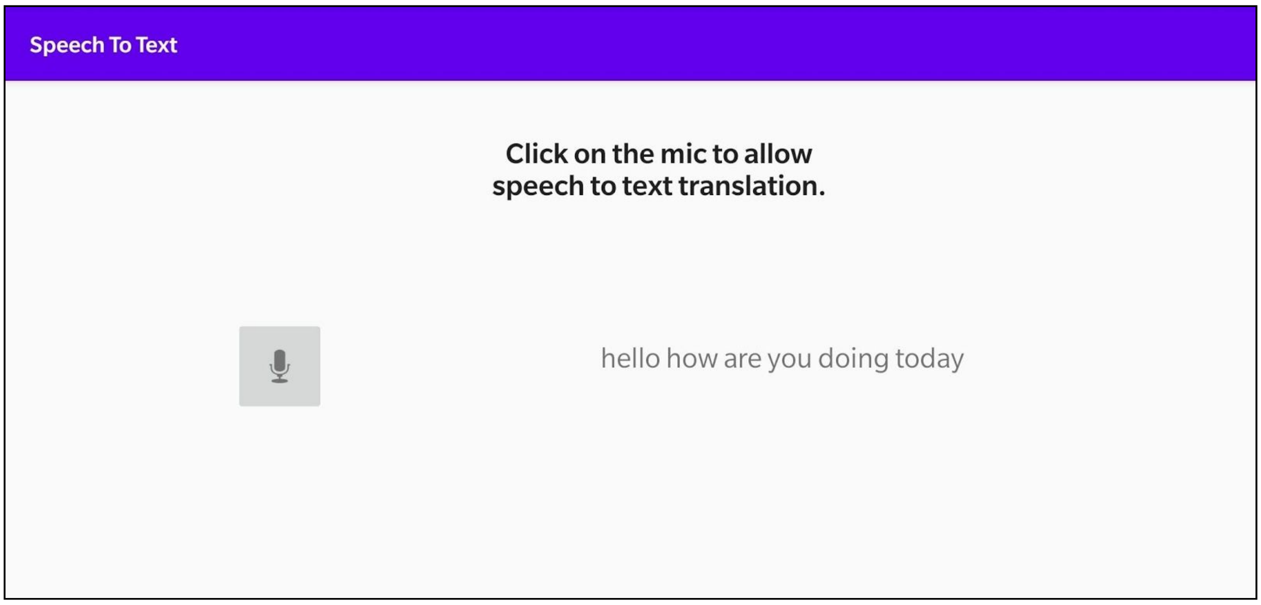

Speech To Text
Click on the mic to allow
speech to text translation.
hello how are you doing today

Fig 5.7 Translated Text Displayed on Speech Translation Mode

Chapter 6

Conclusion and Future Work

6. Conclusion and Future Work

6.1 Conclusion

The resultant application provides a smooth real-time translation of Gesture-to-Text and Speech-to-Text. While the skin detection happens, it is seen that a great deal of noise is generated due to the presence of other objects similar to the colour of skin. This greatly hinders the gesture recognition as the training data used is relatively small and absent of noise.

The quick transition between the 2 modes allows for bi-directional communication. Although the application just handles alphabet translation of gestures, it still provides an effective and fast alternative communication medium for disabled and abled peoples.

6.2 Future Work

There is scope for improvement in the Software Development section. The current proposed model only supports Android platform for mobile devices, this can be extended for other platforms such as iOS. The UI of the application can be scaled and improved to add more functionality.

The methodology can be upgraded as well. The current method makes use of static images of alphabetical hand signs for training and prediction which may be time consuming. An improvement can be made to accommodate dynamic hand gestures in the form of video input where words are directly predicted instead of the need to spell the word letter-by-letter.

Another mode of Translation can be developed - Text to Gestures / Animations. In this case the abled user would be able to communicate with a disabled person without having the need to know sign language.

7. Bibliography

[1] Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil (2019). Sign Language Interpreter using Image Processing and Machine Learning.

[2] Kshitij Bantupalli, Ying Xie (2019) American Sign Language Recognition Using Machine Learning and Computer Vision.

[3] Abhijith Bhaskaran K, Anoop G Nair, Deepak Ram K, Krishnan Ananthanarayanan, H R Nandi Vardhan (2016). Smart Gloves for Hand Gesture Recognition (Sign Language to Speech Conversion System)

[4] Radici, E., Bonacina, S., & De Leo, G. (2016, August). Design and development of an AAC app based on a speech-to-symbol technology. In 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 2574-2577). IEEE.

[5] Rajaganapathy. S, Aravind. B, Keerthana. B, Sivagami. M, Conversation of Sign Language to Speech with Human Gestures ,Procedia Computer Science 50 (2015) 10 – 15

[6] Hongwu Yang, Xiaochun An, Dong Pei, Yitong Liu, (2014), Towards realizing gesture-to-speech conversion with a HMM-based bilingual speech synthesis system.

[7] V. Padmanabhan, M. Sornalatha, (2014), Hand gesture recognition and voice conversion system for dumb people, International Journal of Scientific & Engineering Research, Volume 5, Issue 5, May-2014 ISSN 2229-5518

[8] Joyeeta Singha, Karen Das, Recognition of Indian Sign Language in Live Video (2013). International Journal of Computer Applications (0975 – 8887) Volume 70–No.19, May 2013

[9] Janice Light & David McNaughton (2012). The Changing Face of Augmentative and Alternative Communication: Past, Present, and Future Challenges, Augmentative and Alternative Communication.

[10] Lisa A. Wood, Joanne Lasker, Ellin Siegel-Causey, David R. Beukelman, and Laura Ball (1998). Input Framework for Augmentative and Alternative Communication

[11] Zangari C., Lloyd L. & Vicker B. (1994). Augmentative and alternative communication: An historic perspective. Augmentative and alternative communication, 10(1), 27-59

[12] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5- way-3bd2b1164a53

[13] https://www.tensorflow.org/api_docs/python/

[14] https://keras.io/api/

[15] https://www.tensorflow.org/api_docs/python/tf/lite

[16] https://docs.opencv.org/2.4/modules/refman.html

[17] https://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html

[18] https://cloud.google.com/speech-to-text/docs

[19] https://flaticon.com/