**SIGN LANGUAGE TRANSLATOR USING**

**GESTURE RECOGNITION**

Minor project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

# Computer Science and Engineering

By

## Piyushika Sachdeva 191217

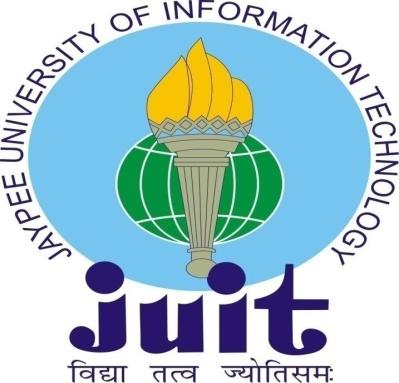
## Mukund Soni 191372

## Aditi Garg 191379

**UNDER THE SUPERVISION OF**

Prof. Dr. Vivek Kumar Sehgal

HOD, Dept of CSE/IT



Department of Computer Science & Engineering and Information Technology

# Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

**TABLE OF CONTENT**

| **Title** | **Page No.** |
| --- | --- |
| **Declaration** | **I** |
| **Certificate** | **II** |
| **Acknowledgement** | **III** |
| **Abstract** | **IV** |
| **Chapter-1 (Introduction)** | **1** |
| **Chapter-2 (Feasibility Study, Requirements Analysis and Design** | **5** |
| **Chapter-3 (Implementation)** | **11** |
| **Chapter-4 (Results)** | **22** |
| **References** | **24** |

**DECLARATION**

I hereby declare that this project has been done by me under the supervision of Prof. Dr. Vivek Kumar Sehgal, HOD, Dept of CSE/IT**,** Jaypee University of Information Technology. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

**Supervised by:**

**Prof. Dr. Vivek Kumar Sehgal**

Professor and Head

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

**Submitted by:**

**Piyushika Sachdeva (191217)**

Computer Science & Engineering Department

Jaypee University of Information Technology

**Mukund Soni (191372)**

Computer Science & Engineering Department

Jaypee University of Information Technology

**Aditi Garg (191379)**

Computer Science & Engineering Department

Jaypee University of Information Technology

**CERTIFICATE**

This is to certify that the work which is being presented in the project report titled “Sign Language Translator using Gesture Recognition” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **“Piyushika (191217), Mukund Soni (191372), Aditi Garg (191379)”** during the period from January 2022 to May 2022 under the supervision of **Prof. Dr. Vivek Kumar Sehgal**, HOD, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

**Piyushika Sachdeva (191217)**

**Mukund Soni (191372)**

**Aditi Garg (191379)**

The above statement made is correct to the best of my knowledge.

**Prof. Dr. Vivek Kumar Sehgal**

Professor and Head

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat,

**ACKNOWLEDGEMENT**

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor**Prof. Dr. Vivek Kumar Sehgal, Professor and Head,** Department of CSE Jaypee University of Information Technology,Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of “**Machine Learning**” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Prof. Dr. Vivek Kumar Sehgal,,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

**Piyushika Sachdeva (191217)**

**Mukund Soni (191372)**

**Aditi Garg (191379)**

**ABSTRACT**

The deaf and mute community has often been neglected and has trouble conveying their thoughts and communicating with people who do not understand sign language. One might not pay attention to the troubles caused until they are bound to interact or communicate and then end up feeling helpless.

Breaking the communication barrier built up between individuals, sign language recognition is a breakthrough and this project aims at making people understand each other despite the communication barrier. This is done by real time recognition of hand gestures of ASL language letters using CNN as the deep learning algorithm, OpenCV, Keras, Tensorflow, Android studio along with other libraries.

The android application captures hand gestures in real time and recognises the gesture translating it into text and further that text into speech. Such applications can be of great help to children who are deaf and mute as well as their abled siblings to learn sign language much more interactively.

With this aim in mind, this project is a step towards achieving inclusion and making communication a little easier when it comes to people who speak sign language or wish to learn it.

**Chapter 1: INTRODUCTION**

* 1. **Introduction**

**Sign Language**

Sign language (SL) [1] is a visual-gestural language used by deaf and hard-hearing people for communication purposes. Meanings are conveyed by hand movements and gestures with their own vocabulary and syntax.

Gesture recognition system framework comprises a simple, effective and accurate mechanism to change gestures into text or speech. Fundamental parts of sign language recognition are data acquisition, data preprocessing and transformation, feature extraction, classification and finally the result (into text or speech).

Communication via gestures acknowledgment separating the correspondence obstruction among people and looking back Sign language recognition targets breaking down the communication barrier among people and offers more chances to the hard of hearing and quiet to be heard and perceived consistently.

**CNN**

Computers cannot see things as we do, for computers image is nothing but a matrix. The Input to a layer in CNN is 3 dimensional containing height, width and depth. It basically comprises Convolutional layer, pooling layer and fully connected layer and then the final output layer.

**1. Convolution Layer :**

Convolution layer, is the building block of CNN [8], and essentially is a layer in which objects convolve on one another. The learnable filter is one of the matrices which convolves over a restricted portion of the image represented as another matrix and results in a 2D activation matrix. There is an activation matrix by the end of the convolution process with fewer parameters(dimensions) than the actual image as well as more clear features than the actual one.

**2. Pooling Layer :** A pooling layer [8] is used to further decrease the size of the activation matrix and ultimately reduce the learnable parameters as well as the computational power to process the data. The two types of pooling are:

**a) Max Pooling :** In this approach, we take the maximum amongst all the numbers which are in the pooling region(kernel) and shifting the pooling region each time to process another neighborhood of the matrix(input).The final output is a matrix half the size of activation matrix.

**b) Average Pooling :** In average pooling, we take the average of all values in the pooling region.

**3. Fully Connected Layer :** Until now, certain features of the image were extracted and its dimensions were reduced. This layer marks the beginning of the classification of images.

Now that the input has been appropriately transformed, it is flattened into one column vector and fed into the feed forward neural network and backpropagation is applied to every iteration when training the data.

**4. Final Output Layer :** The output of a fully connected layer [8] is fed into a final layer of neurons[having count equal to total number of classes] which generates the final output in the form of probability i.e values between 0 and 1 for the final prediction.

* 1. **Objective**

Sign language is a visual language and consists of 3 major components :

**Fingerspelling:** Used to spell words letter by letter.

**Word level sign vocabulary:** Used for the majority of communication.

**Non- manual features:** Facial expressions and tongue, mouth and body positions.

With this in mind, our project targets at making an android application to recognise the Fingerspelling static hand gestures, combine those alphabets to form words and then translate those words from text into speech.

Following is an image of the ASL gestures that we aim to train:

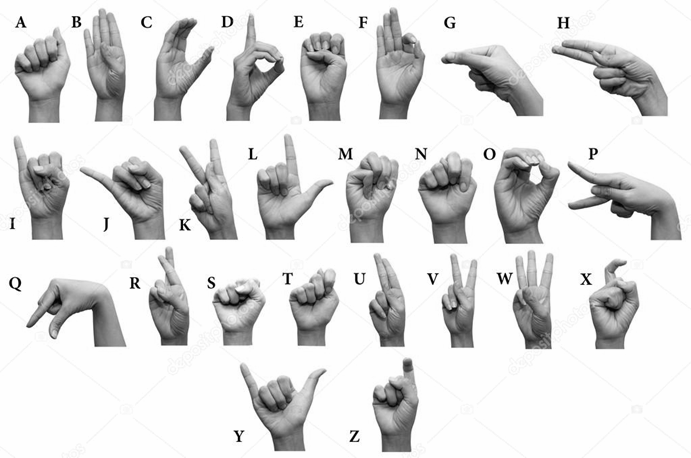


Fig 1. ASL SYMBOLS

* 1. **Motivation**

With so many people relying on sign language to communicate, it becomes almost impossible to hold a meeting where a disabled person and an abled person cannot communicate and help each other. It can also be noted that, unlike spoken languages, which vary greatly from country to country and are based on different regions, more than 80% of people with disabilities can communicate and understand through sign language, despite belonging to completely different regions. Therefore, a work project aimed at breaking down this communication barrier would make a great contribution to the betterment of people with disabilities.

* 1. **Language Used**

**Python**

**Python** is the scripting language used in the creation of this project. Several python libraries commonly used in building neural networks were used, such as Keras and TensorFlow.

**Keras**

Keras is used when a neural network needs to be quickly built and executed in a few lines of code and is a wrapper for tensorflow. It contains implementations of commonly used neural network elements such as layers, targets, activation functions, optimizers, and tools to make working with images and text data easier.

**Tensorflow**

Tensorflow is the library responsible for the mathematical computation behind a neural network which includes computing and optimizing the loss function by adjusting the weights and biases in order to get accurate prediction results.

**OpenCV**

It is an open source library of functions used mainly for image processing, object recognition, feature analysis and capturing video or images.

**Android Studio**

Android Studio is the official Integrated Development Environment (IDE) for Android app development. It contains all the Android tools to design, test, debug, and profile your application.

* 1. **Technical Requirements ( Hardware)**
* A mobile phone is of utmost importance in this project in order to run the android application on.
* An integrated camera in a phone is one of the most important hardware requirements.
* GPU of minimum 4 GB ram is required for training the model efficiently.
  1. **Deliverables/Outcomes**
* Prediction of class of alphabets the input belongs to
* Conversion of predicted text into speech
* Accuracy measure of the prediction

**Chapter 2: Feasibility Study, Requirements Analysis and Design**

**2.1 Feasibility Study**

**2.1.1 Problem Definition**

* The problem can be defined as identifying the hand in an image.
* Furthur the image dataset needs to be preprocessed to reduce noise and improve results.
* Identify what the hand gesture means using a neural network and labeled dataset.
* Classify the given input as one of the twenty six alphabets of ASL and display the text in the application.
* Convert the predicted class of gesture in the form of text into speech.

**2.1.2 Problem Analysis**

* Train a neural network model using the preprocessed dataset to classify any given input into one of twenty six classes (representing ASL alphabets).
* Import the model to android studio and capture any hand gesture real time.
* Recognise the captured gesture by classifying it into an output class and translate it into text.
* Convert the text to speech.

**2.1.3 Solution**

* Using OpenCV, the hand is identified from the image using image segmentation on the basis of hand color.
* Using CNN, and the dataset along with target variable values as the classes of alphabets, a neural network is created to be fed the input image from the phone’s camera through the app.
* Converting the trained model into a tflite model to work in an android application.
* Using pre trained model to convert text to speech

**Methodology**

Lately there has been colossal research done on recognising gestures. Utilizing Hidden Markov Models (HMMs), resulted in limiting recognition of context dependent models. ASL is one of the most widely recognized dialects for creating communication through signing.

With the assistance of literature review done, the basic steps in creating recognising gestures are :

1. Data acquisition

2. Data preprocessing

3. Feature extraction

4. Gesture classification

**Data acquisition:**

**1. Sensor based**

Electromechanical devices are used to extract hand motion, configuration and position. Different glove based approaches can be used to extract information . But it is expensive and not user friendly.

**2. Vision based**

In vision-based technique, to gather the information from hands or fingers, a computer camera is used as the- input device. Acquired data is either a single frame of images or continuous frames of images. The main challenge of vision-based hand detection is coping with the large variability in the appearance of the human hand due to a large number of hand movements, different skin color possibilities, as well as the variations in the points of view, scales and speed of the camera capturing the scene.

**For vision based approach:**

**Data preprocessing and Feature extraction:**

By using filters like Grayscale, performance can be increased by reducing noise in images.

● In [9], the approach to hand detection combines threshold-based color detection with background subtraction.

● Another way to get data (from video extraction) more accurately than using filters is by using instrumented gloves and also hence reduce computation time.

●Segmentation is the process of partitioning images into several distinct parts. In this phase, a region of interest (ROI) is segmented from the rest of the image [3]. Skin color segmentation takes many dependencies such as camera properties, lighting, skin color into account. This can lead to poor accuracy, and to improve this we chose to keep the hand's background a single color to eliminate the need to segment it based on skin color.

**Literature Survey**

**Byeongkeun Kang , Subarna Tripathi , Truong Q. Nguyen(2015**)[1]-The authors talk about using depth sensors in capturing additional information.Along with the use of GPU and CNN along with depth sensor approach, accuracy is increased. Hand segmentation is done by first considering the closet part of image from the camera as the hand along with using balck wristband which creates depth void around wrist. By finding connected components from the closest region in this depth image hand segmentation is achieved.

**Daniels, M. (1994)**[2]-Emphasizes the better performance of children belonging to either language delayed or disordered population or the hearing kids of deaf parents if they learn sign language at an early age along with traditional language learning. This was tested by taking 60 students from 2 different schools and teaching children of one school traditional instructions while the other sign language as well. Children when taught sign language at early age tend to learn it at a commendable rate with no effect on the learning rate of traditional languages.

**Han, J., Awad, G., & Sutherland, A. (2009)**[3]- Discussion of a hand segmentation model that can handle variations in skin color by authors. Skin segmentation is based on a support vector machine model and consists of a training stage and a segmentation stage. A generic skin color model is applied to multiple frames in the training stage showing the initial skin areas. Subsequently, an SVM active learning-based binary classifier is trained using the obtained initial skin pixels. In the segmentation stage, the SVM classifier is embedded with region information to produce the final skin color pixels.

**Mahesh, M., Jayaprakash, A., & Geetha, M. (2017, September)**[4]- The authors proposed an Android application in which a gesture can be added as part of a data set and the gesture recognized. Gestures are primarily identified by descriptors and then, for the comparison to be made by descriptors, the data set is reduced by histogram comparison. For the recognition part, the image taken is first preprocessed. The images present in the dataset are loaded one by one at each iteration. The uploaded images are pre-processed afterwards. The two pre-processed images are fed to the matcher which uses histogram matching and ORB descriptor matching to compare the images and give the image name as output if a good match is found.

**Ng, C. W., & Ranganath, S. (2002)**[5]- In this work, real time gesture recognition is applied to manipulate windows and objects in GUI. For hand segmentation, first binary hand blobs are extracted followed by application of fourier descriptors which shape the blobs. This is given as input to the radial basis function for classification. HMMs and RNN are investigated and best results were observed when HMM and RNN were combined.

**Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015)**[6]- The authors proposed to use Microsoft Kinect along with CNN and GPU to develop a recognition system since CNNs automate the process of feature construction instead of constructing complex handcrafted features. They recognised 20 Italian gestures with a cross validation accuracy of 91.7%.

**Tan, M., & Le, Q. (2019, May)**[7]-A new model scaling method is discussed here by the authors wherein it is acknowledged that by making models too wide,deep or of very high resolution, the model saturates quickly and ends up being not very efficient and merely a large number of parameters. By scaling the model gradually (Refer Fig 2) whether it be width,depth or higher resolution, the fewer number of parameters contribute to a better and efficient result.

**Wu, J. (2017)**[8]- The authors here explain the basics of CNN which include the convolution layer that is the building block of CNNs. Activation functions along with the most widely used activation function, Rectified Linear Unit (ReLU) has also been described in this work. The ReLU function is used to add non linearity to the data. Pooling layer is also an important element in CNN which decreases the size of activation matrix and is of two types namely, average and max pooling.

**Yang, J., & Xu, Y. (1994)**[9]-In it Hidden Markov Models (HMM) are used to classify gestures. This model deals with the dynamic aspects of gestures. Gestures are extracted from a sequence of video images by tracing the patches of skin color corresponding to the hand in a body-face space centered on the user's face. The goal is to recognize two types of gestures: deictic and symbolic. The image is filtered using a quick index array. After filtering, skin color pixels are collected in blobs. Blobs are statistical objects based on the position (x, y) and colorimetry (Y, U, V) of skin color pixels to determine homogeneous areas.

**Ziaie, P., Müller, T., Foster, M. E., & Knoll, A. (2008, March)**[10]- Naive Bayes Classifier is used in this work which is an efficient and fast method of static hand gesture recognition. It is based on the classification of different gestures according to geometry-based invariants, which are obtained from the image data after segmentation. Unlike many other detection methods, this method is independent of skin color. Gestures are extracted from each frame of the video with a static background. The first step consists of segmenting and labeling the objects of interest and extracting geometric invariants from them. The next step is to classify the gestures using a K-Nearest Neighbor algorithm compatible with a distance weighting algorithm (KNNDW) to provide data suitable for a locally weighted Naive Bayes classifier.

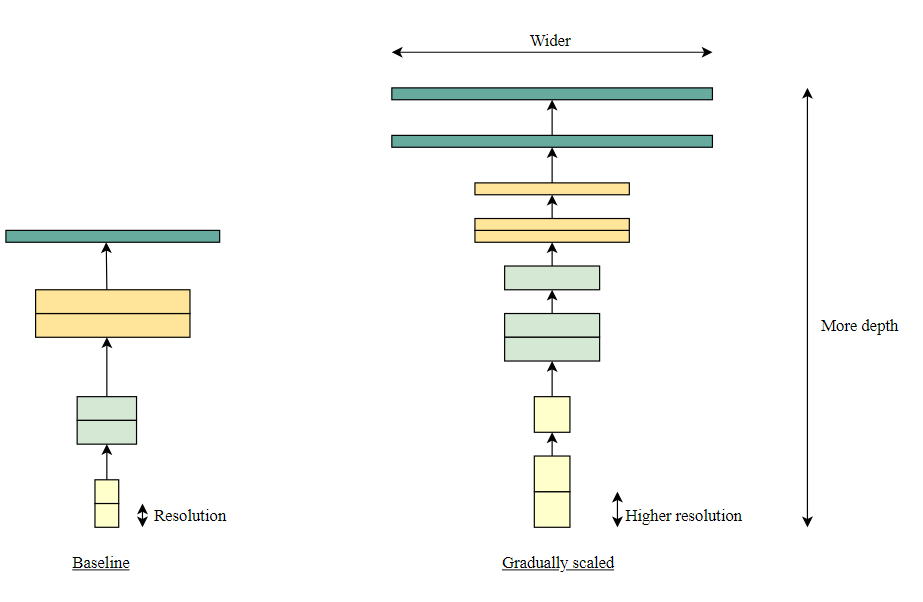


Fig 2: Model Scaling in EfficientNet

**2.2 Requirements**

**2.2.1 Functional Requirements**

* The Application can Recognize Different ASL Symbols.
* The Application can make words from recognised text.
* The Application can Speak the Recognised text as a Word.

**2.2.2 Non-Functional Requirements**

* Model can only detect Static symbols.
* The Symbols that use motion can’t be recognised by Application.

**2.3 E-R Diagram / Data-Flow Diagram (DFD)**

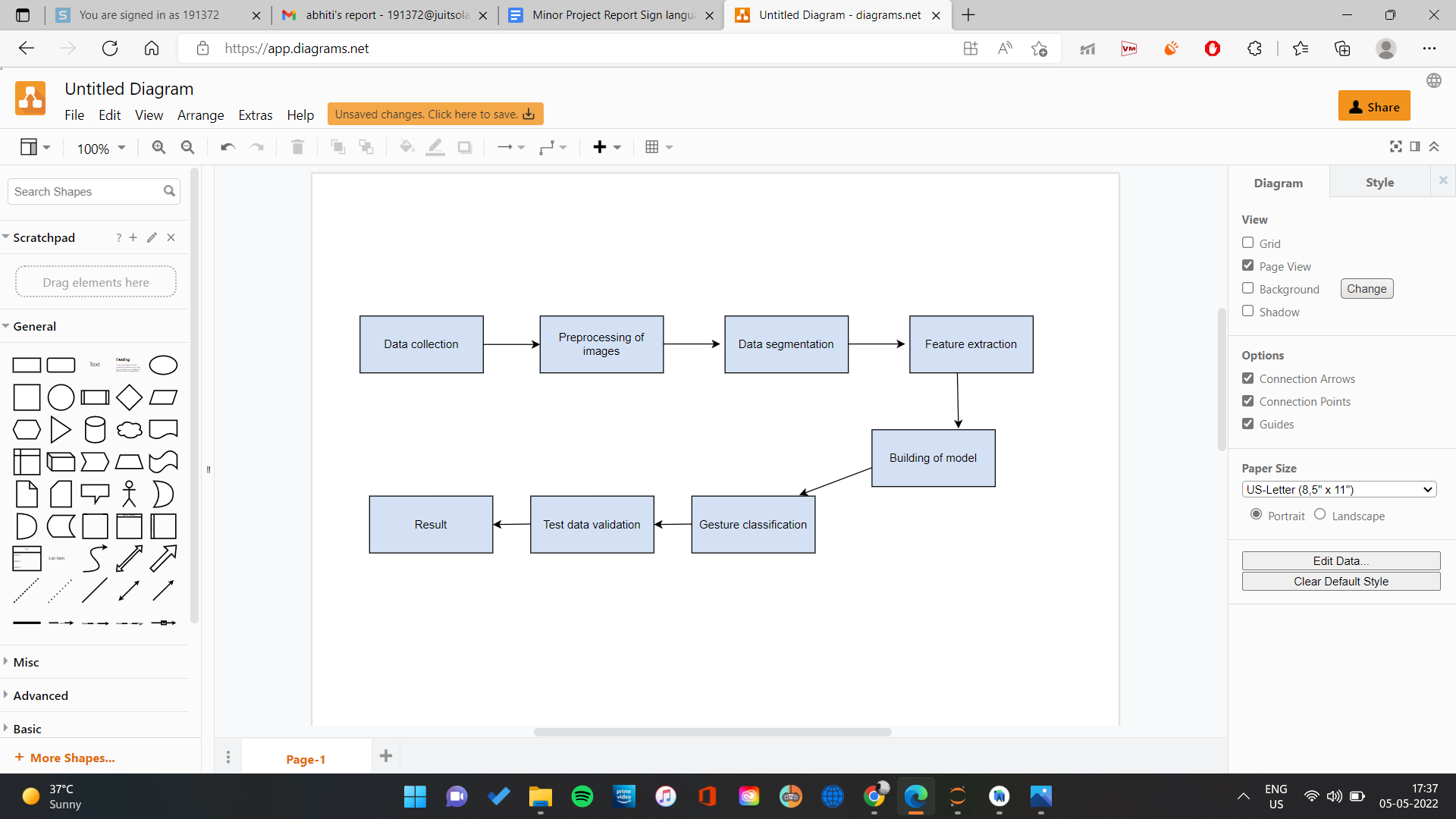


Fig 3.1 Flow of data during model creation

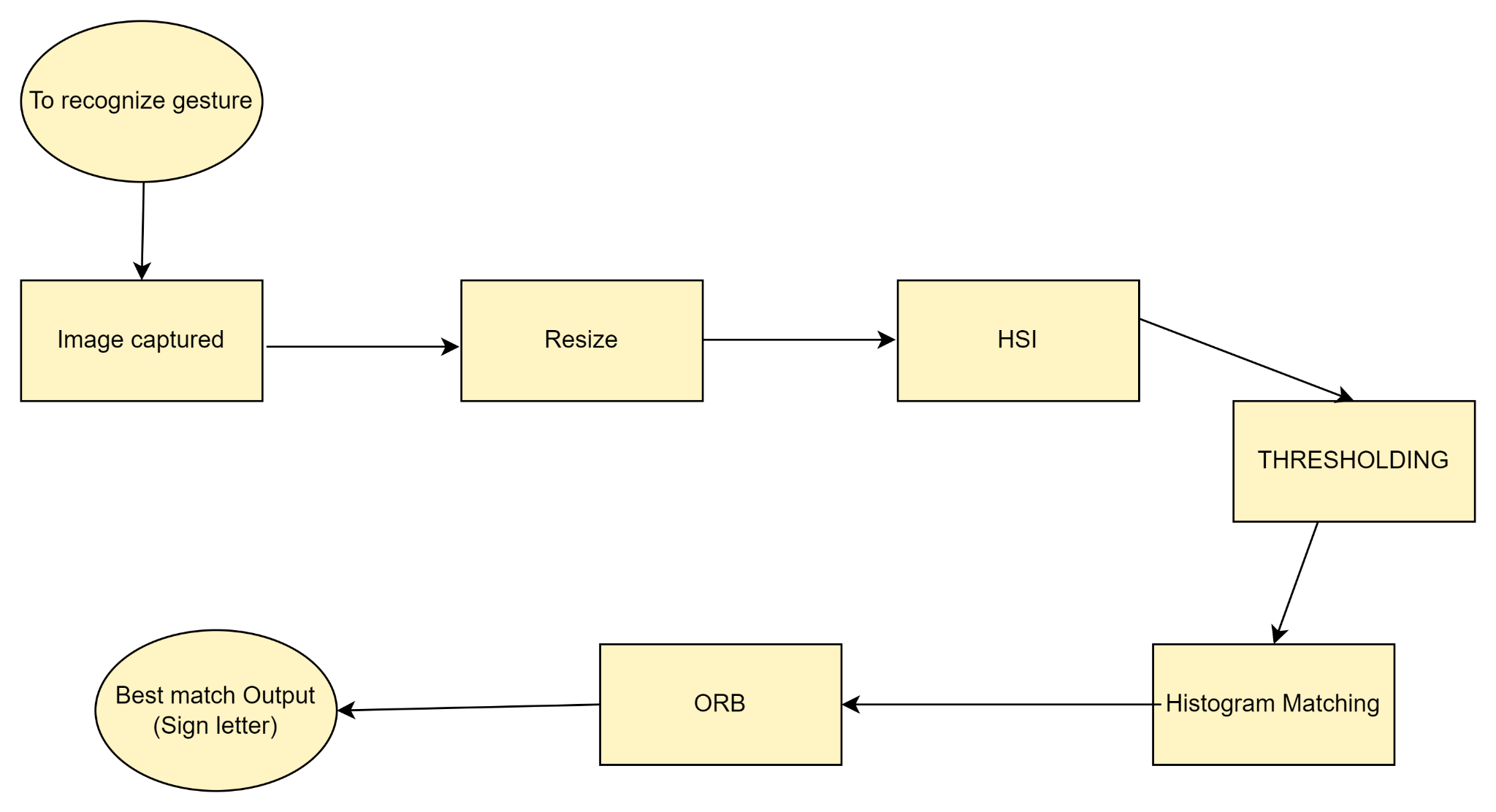


Fig 3.2 Flow of data in android application

**Chapter 3: IMPLEMENTATION**

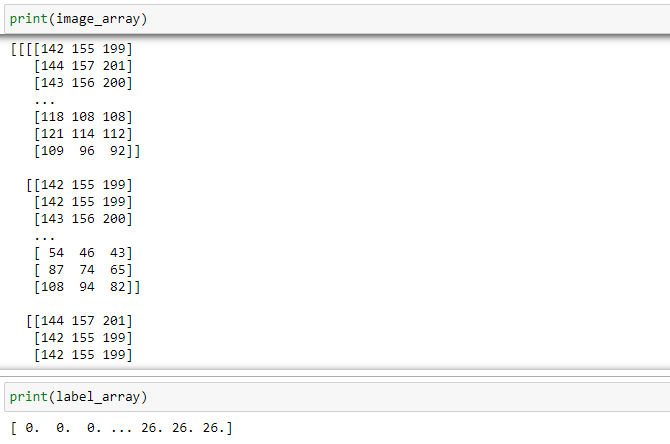
**3.1 Date Set Used in the Minor Project**

For this project, the dataset was preprocessed by converting the captured raw RGB images of the hand gesture lying in the ROI into Grayscale image, also resizing the images as per our requirements. The algorithm is trained using a set of about 800 images per letter and testing is done on a completely different set of data (200 images). The OpenCV library was used to perform these operations.

**3.2 Date Set Features**

**3.2.1 Types of Data Set**

* **Image Array and Label Assigned to that Array**

****

**3.2.2 Number of Attributes, fields, description of the data set**

* **300-400 colored images per symbol of each letter from A to Z.**

**3.3 Design of Problem Statement**

* **Designing of Model**
  + Model should be compatible with Android Studio
  + High Accuracy in Recognizing Image

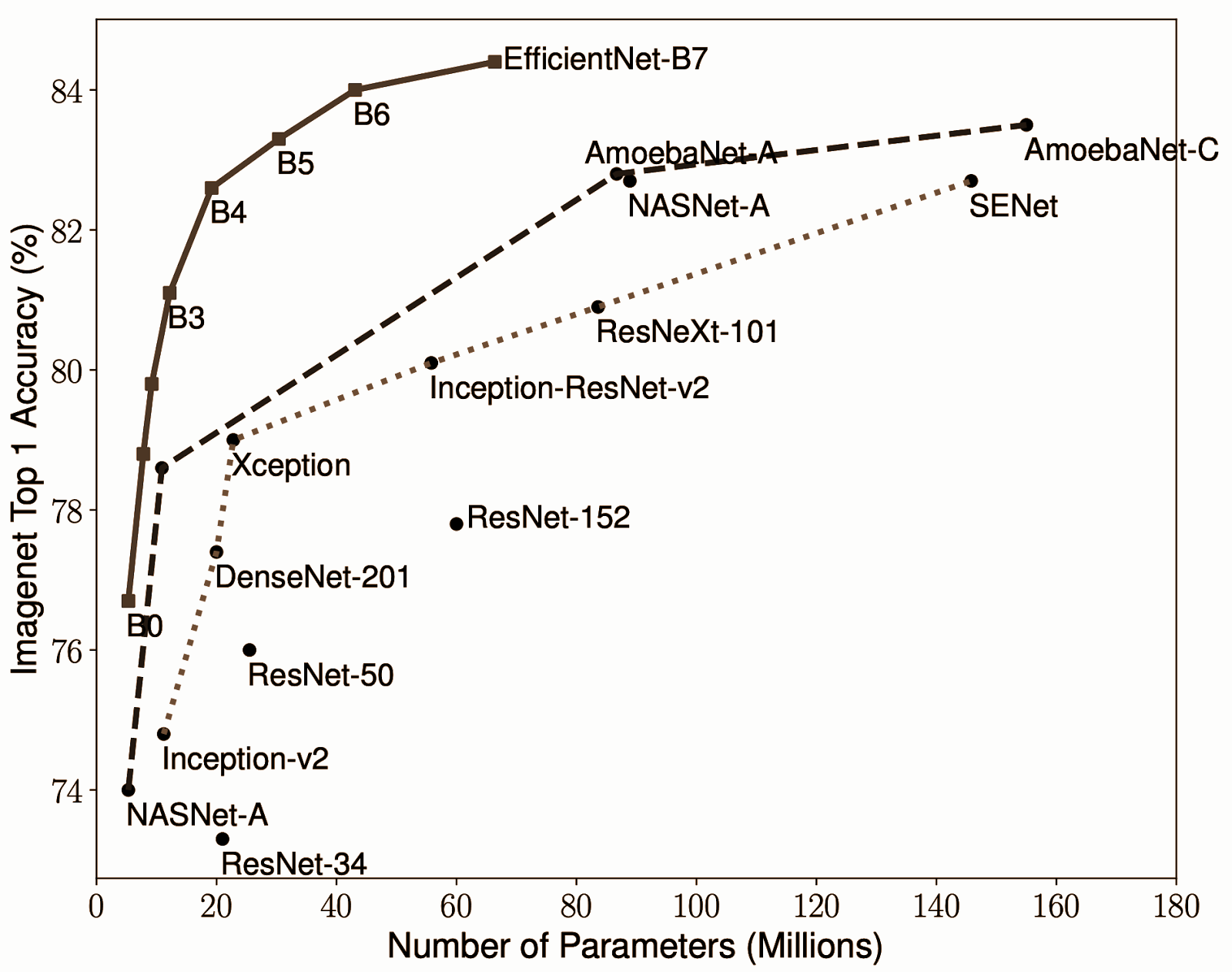


Fig 4. Different EfficientNetB0 - EfficientNetB7

* **Integrating in Android Studio**
  + Finding Compatible OpenCV Module for Android studio
  + Accessing Camera and Converting it to Bitmap such that Recognition becomes Possible
  + Integrating Model to Recognize Image on screen
  + Using that recognized Letter and Forming a String
  + Converting String to Speech
* **Designing the Layout**
  + Apk Style (To keep it Simple)
  + Color Schemes

**3.4 Algorithm / Pseudo code of the Project Problem**

**Model used: EfficientNet B4 along with addition of average pooling, dense and a dropout layer.**

For the model:

* Data Preprocessing step.
* Data in label arrays and then split them in train and test.
* Then using ksera import modules.
* Create a model and add an ‘adam’ optimizer.
* Test accuracy of the result.
* Visualizing the test set result

For Android studio:

* Firstly, integrate OpenCV.
* Use the camera module in it and give permission for the camera.
* Integrate Model in Android Studio.
* Then use OpenCV to make camera images into bitmap.
* While creating bitmap use interrupt and assign it number of threads.
* In bitmap detect and recognize the image.
* Then use that recognized data to make a word.
* Add text to speech function from android studio to speak the word created.
* To do these operation assign buttons

**3.5 Flow graph of the Minor Project Problem**

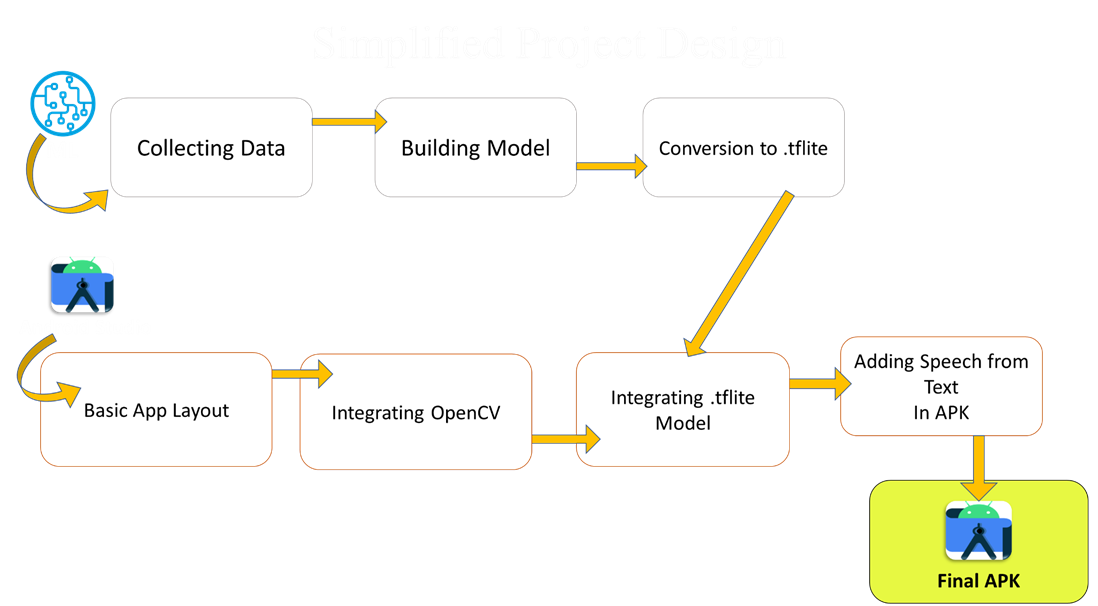
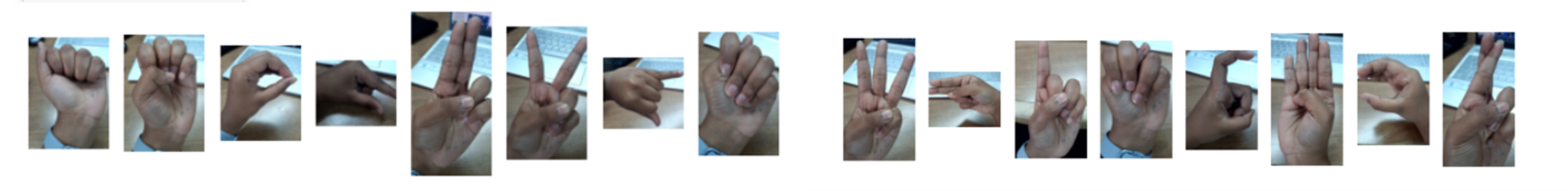
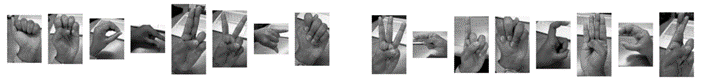
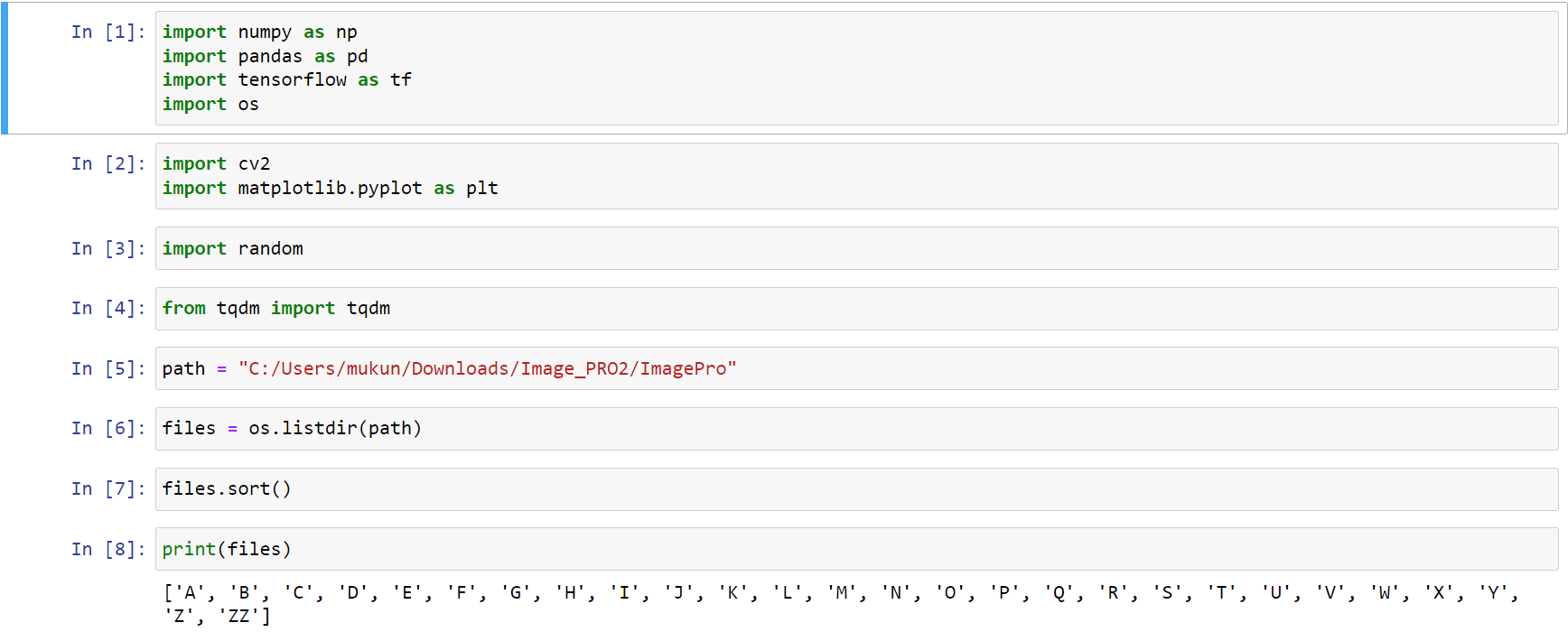
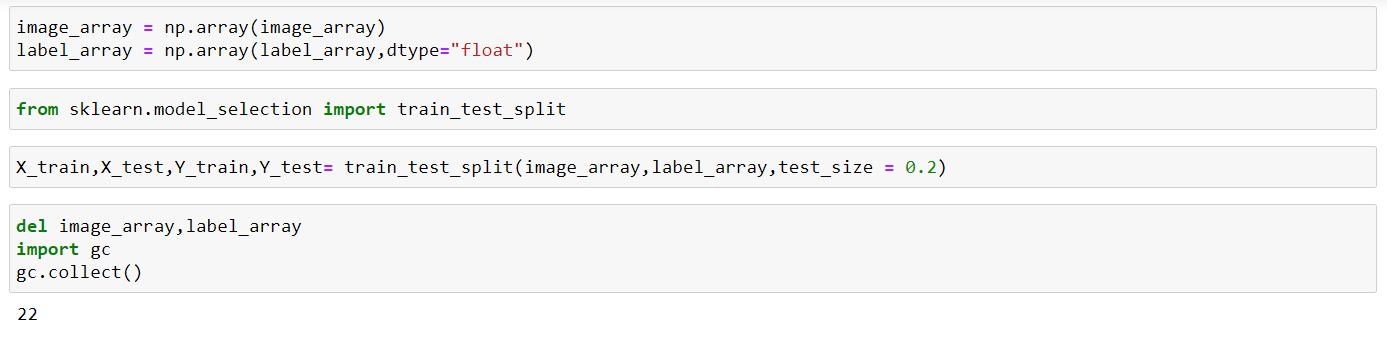
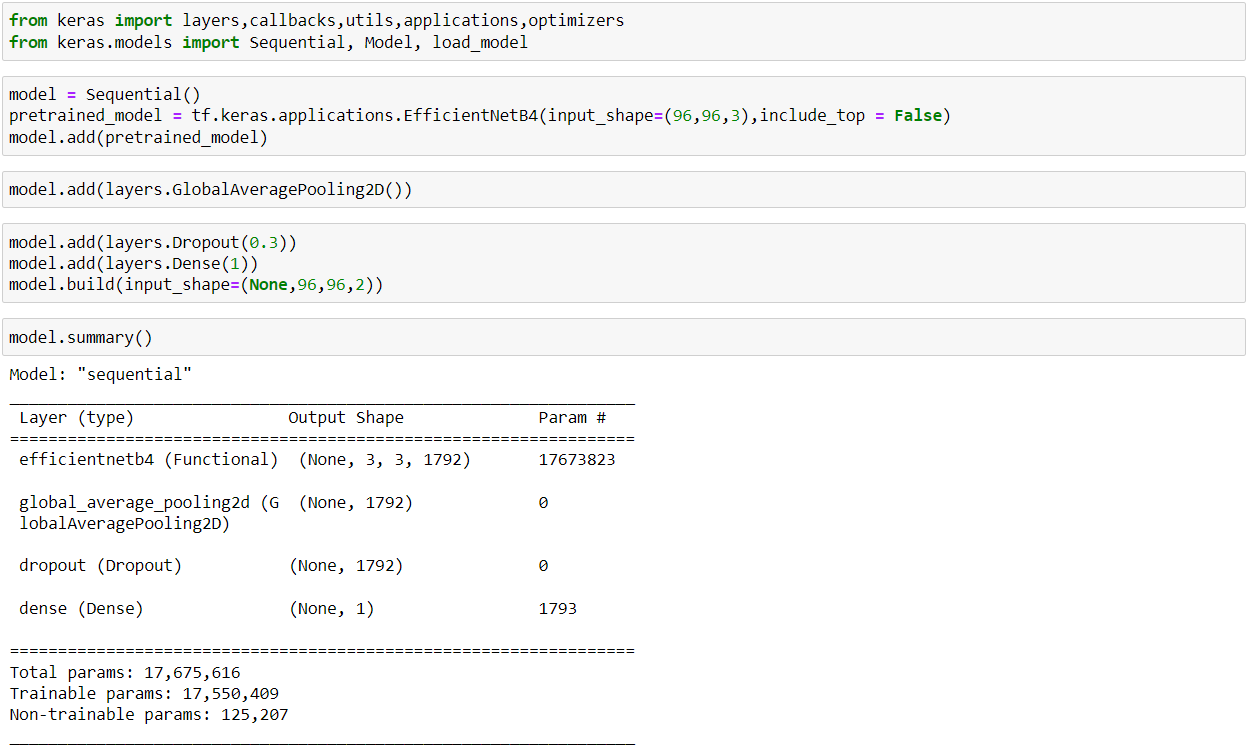
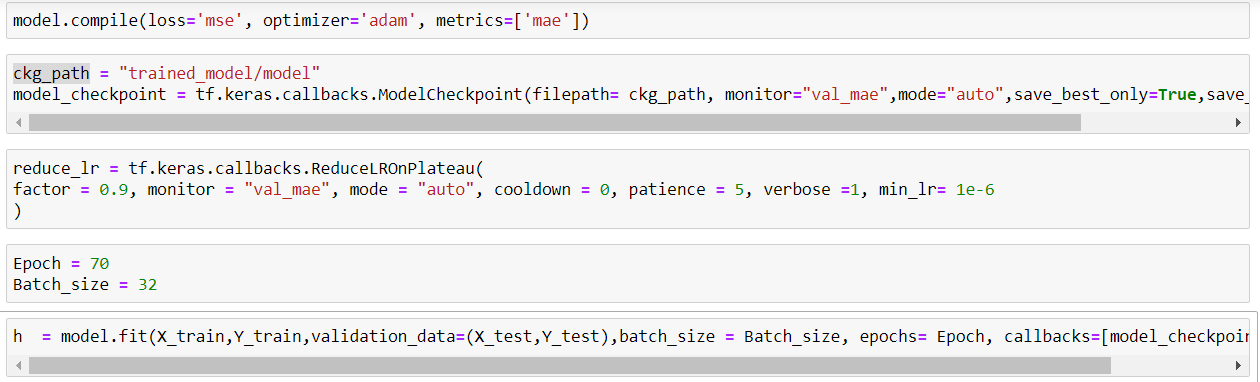
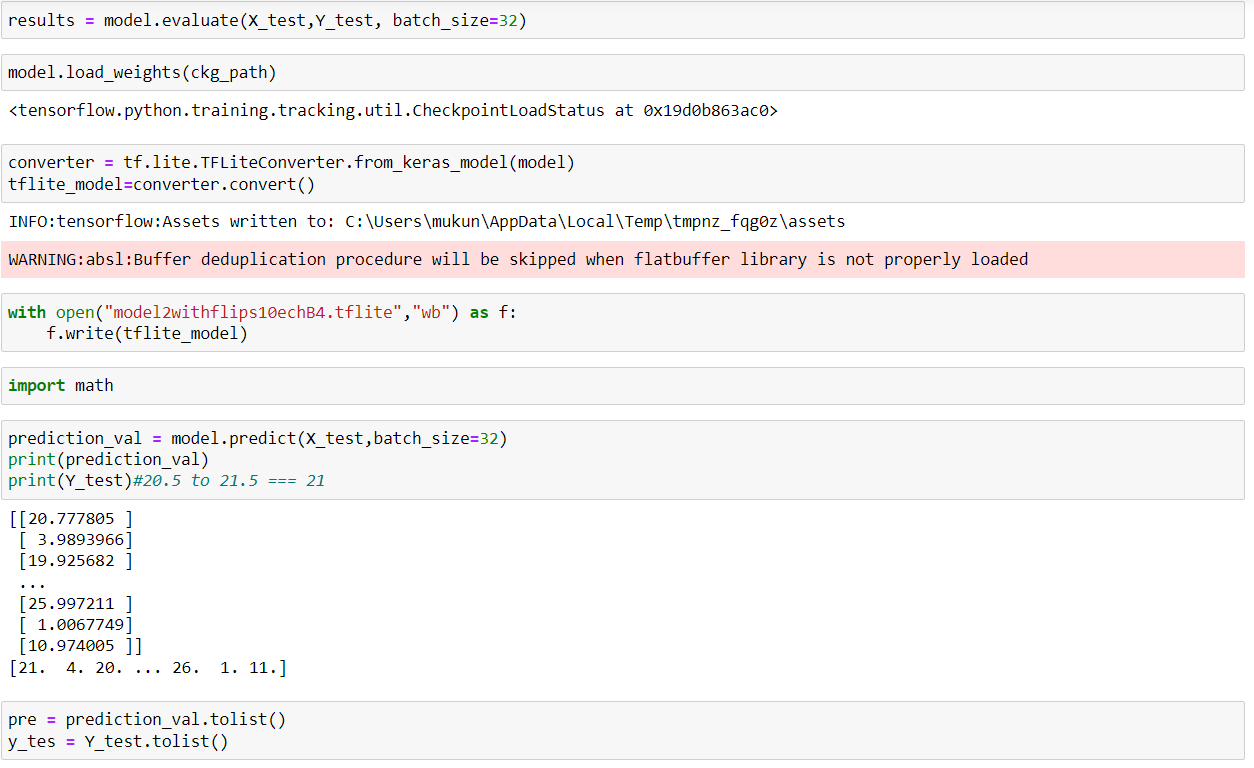
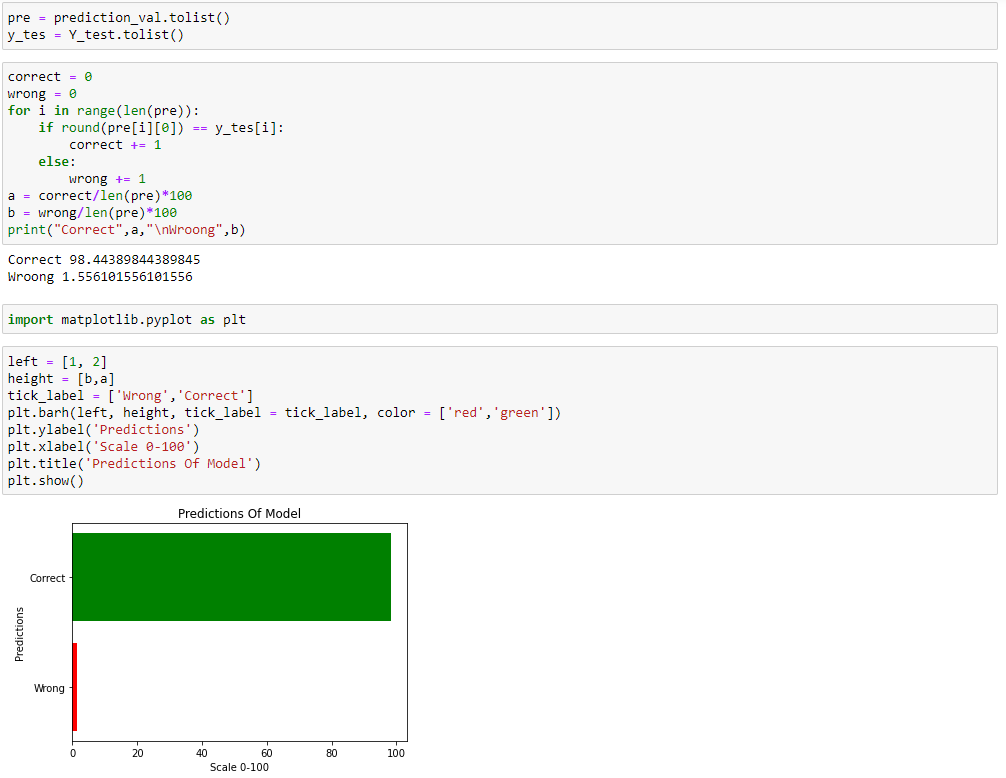
****

Fig 5. Flow Graph of Minor Project

**3.6 Screenshots of the various stages of the Project**

1. **DataSet Building Collecting Images to Process**
   1. 
   2. 
2. **Model Building** 
   1. Imported Modules and Defined DataSet Location 
   2. Created 2 Labels arrays to store data and used OpenCV to Extract the Data like:BGR to RGB Image conversion data.
   3. Dividing the Data in Test And Train Also Cleaning the Arrays
   4. Building the Model 
   5. Processing the Model
   6. Resultant Model and Its Evaluation





**3. Designing Layout of APK**

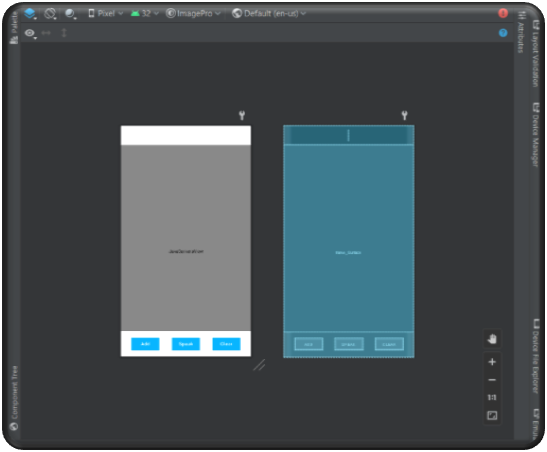
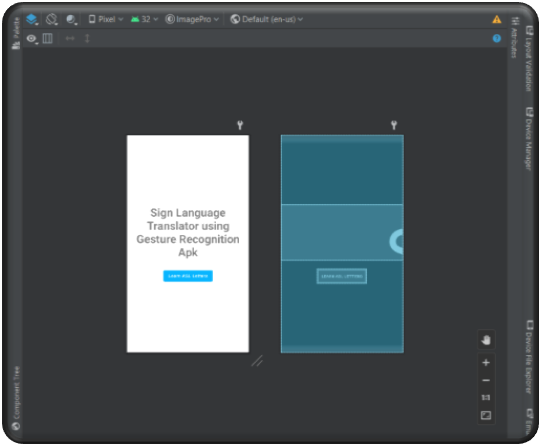


Fig 6. activity\_main.xml Fig 7. activity\_combine\_letters.xml

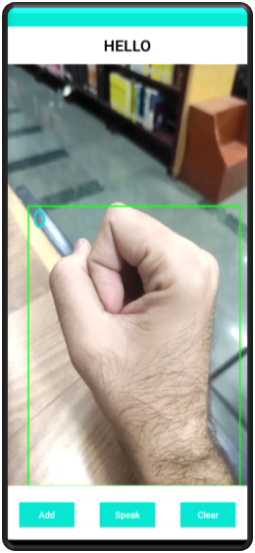
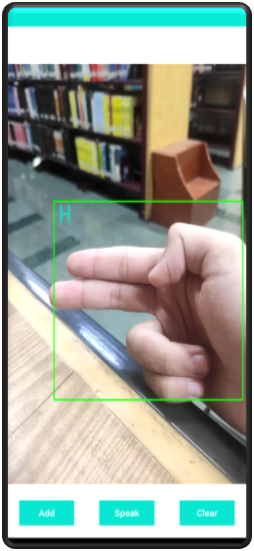


Fig 8.Sample Images of APK

**4. Pseudo Code for Android Application**

* MainActivity.java
  + begin

if (Opencv Loaded)

log(“Opencv Integration Successful”)

else

log(“failed to load OpenCV”)

create button;

onbuttonclick{

startActivity CombintLettersActivity.java

};

end

* CombintLettersActivity.java
  + begin

Create Mat mRgba;

Call CameraBridgeView from OpenCVCameraView

Call SignLanguageClass from SignLanguageClass.java

Create buttons add,clear and speak;

Create Textview change\_text;

Baseloader{ Load MOpenCvCameraView; }

OnCreate{ Ask Camera Permissions;

Call SignLanguageClass;}

OnResume{ Load OpenCv;

Call mloaderCallback;}

OnPause{ disable MOpenCvCameraView;}

OnDestroy{ disable MOpenCvCameraView;}

OnCamareViewStarted(width,height){

mRgba = New Mat(height,width,CvType.CV\_8UC4);

}

OnCamerVIewStopped(){

mRgba.release();

}

Mat OnCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputframe){

mRgba = inputframe.rgba();

Mat out = new Mat();

out = SignLanguageClass.recognizeImage(mRgba);

return out

}

end

* SignlanguageClass.java
  + begin

Create Interpreter interpreter, interpreter2;

Create list<string> Lablelist;

Create GpuDelegate gpudeligate;

int height, width = 0;

int Classification\_Input\_Size = 0;

int Input\_Size;

string final\_text, current\_text = “ ”;

Create TextToSpeech texttoSpeech;

SignLanguageClass (){

Input\_Size = inputsize;

Classification\_Input\_Size = classification\_inputsize;

#create Interpreter option

Interpreter.options options = new Interpreter.Options();

Interpreter.options options2 = new Interpreter.Options();

#Assign GPU and Threads

gpuDelegate=new GpuDelegate();

options.addDelegate(gpuDelegate);

options.setNumThreads(4);

#Assign Threads to Recognizer

option2.setNumThreads(3);

interpreter2=new Interpreter(loadModelFile(assetManager,classification\_mode),option2);

Add buttonOnClick{

final\_text = final\_text+current\_text;

change\_text.setText(final\_text);

}

Clear buttonOnClick{

if(final\_text.length()!=0){

final\_text = final\_text.substring(0, final\_text.length() - 1); change\_text.setText(final\_text);

}

}

Create texttoSpeech = new TexttoSpeech(){

texttoSpeech.setLanguage(Locals.US);

}

Speak buttonOnClick{

textToSpeech.speak(final\_text,TextToSpeech.QUEUE\_FLUSH,null);

}

LoadLablesList(){

Create String list;

while((line=reader.readLine())!=null){

labellist.add(line);

}

return labellist

}

ByteBuffer loadModelFile(){

Assign openmodelpath;

Create inputStream;

Create fileChannel from InputStream;

Create offset from fileChannel;

Create declaredlength from fileChannel;

return fileChannel.map();

}

Recognizer{

Create Bitmap;

Assign height, width;

Convert BitMap to ByteBuffer through BytetoBuffer;

ByteBuffer byteBuffer=convertBitmapToByteBuffer(scaledBitmap);

Create Box from Recognition;

Interpreter runBuffer Recognize;

string a = get\_alphabets(runBuffer);

}

string get\_alphabets(float){

String val = "";

if ((sig\_v >= -0.5 & sig\_v < 0.5) ) { val = "A";}

else if ((sig\_v >= 0.5 & sig\_v < 1.5) ) {val = "B";}

else if ((sig\_v >= 1.5 & sig\_v < 2.5)) {val = "C";}

else if ((sig\_v >= 2.5 & sig\_v < 3.5)) {val = "D";}

else if ((sig\_v >= 3.5 & sig\_v < 4.5)) {val = "E";}

else if ((sig\_v >= 4.5 & sig\_v < 5.5)) {val = "F";}

else if ((sig\_v >= 5.5 & sig\_v < 6.5)) {val = "G";}

else if ((sig\_v >= 6.5 & sig\_v < 7.5)) {val = "H";}

else if ((sig\_v >= 7.5 & sig\_v < 8.5)) {val = "I";}

else if ((sig\_v >= 8.5 & sig\_v < 9.5)) {val = "J";}

else if ((sig\_v >= 9.5 & sig\_v < 10.5)) {val = "K";}

else if ((sig\_v >= 10.5 & sig\_v < 11.5)) {val = "L";}

else if ((sig\_v >= 11.5 & sig\_v < 12.5)) {val = "M";}

else if ((sig\_v >= 12.5 & sig\_v < 13.5)) {val = "N";}

else if ((sig\_v >= 13.5 & sig\_v < 14.5)) {val = "O";}

else if ((sig\_v >= 14.5 & sig\_v < 15.5)) {val = "P";}

else if ((sig\_v >= 15.5 & sig\_v < 16.5)) {val = "Q";}

else if ((sig\_v >= 16.5 & sig\_v < 17.5)) {val = "R";}

else if ((sig\_v >= 17.5 & sig\_v < 18.5)) {val = "S";}

else if ((sig\_v >= 18.5 & sig\_v < 19.5)) {val = "T";}

else if ((sig\_v >= 19.5 & sig\_v < 20.5)) {val = "U";}

else if ((sig\_v >= 20.5 & sig\_v < 21.5)) {val = "V";}

else if ((sig\_v >= 21.5 & sig\_v < 22.5)) {val = "W";}

else if ((sig\_v >= 22.5 & sig\_v < 23.5)) {val = "X";}

else if ((sig\_v >= 23.5 & sig\_v < 24.5)) { val = "Y";}

else if ((sig\_v >= 24.5 & sig\_v < 25.5)) {val = "Z";}

else ((sig\_v >= 25.5 & sig\_v < 26.5)) {val = " ";}

return val;

}

ConvertBytetoBuffer(){

Create Flag quant = 1;

if quant == 0

allocate(1\*sizeimg\*sizeimg\*3)

else

allocate(4\*sizeimg\*sizeimg\*3)

bitmap getpixels(height,width);

for (int i=0;i<size\_image;++i)

for (int j=0;j<size\_image;++j)

if(quant == 0)

bytebuffer put(byte) (val)

else

bytebuffer put(byte) (val)

}

end

**Chapter 04: RESULTS**

**4.1 Discussion on the Results Achieved**

* The Prediction/Accuracy of the model comes out to be 98.4%
* Final Application Shows the rest of Results.

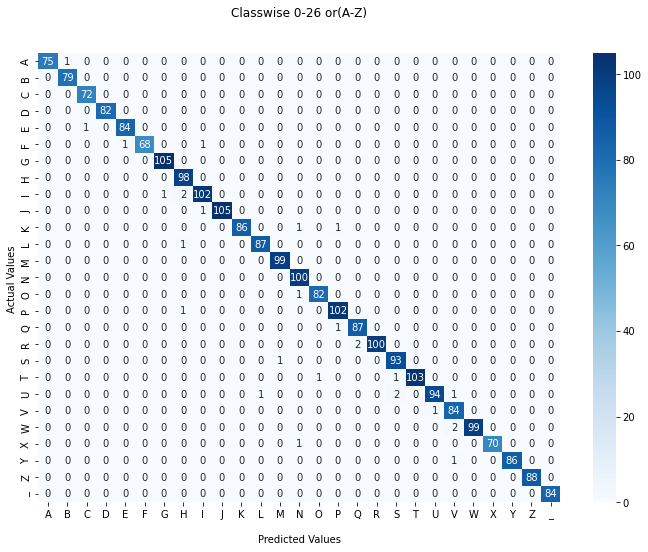
****

Fig 9. Confusion matrix



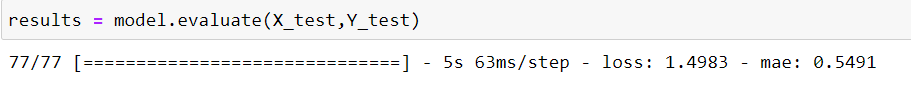


Fig 10. Different Val\_loss after 10 Epoch

**4.2 Application of the Minor Project**

An android application that can recognise sign gestures and moreover translate it into speech can be of great benefit to learning children especially austistic children, hearing children of deaf parents as well as children with siblings who are deaf or language delayed or disordered.

**4.3 Limitation of the Minor Project**

* At this point, the model can detect only static symbols i.e gestures involving no sort of hand motion and hence no video processing. However, the majority of real world conversations among people talking in sign language happens through gestures defined by certain motions. Thus, detecting static gestures is not enough for efficient and seamless conversations in real life.
* Dataset used can be improved by adding images with various different skin color shades to recognise most of them in the android application and with different background and lighting conditions to improve the efficiency of detecting the gestures correctly.

**4.4 Future Work**

Gestures requiring dynamic motion require video processing, which wasn’t a feasible approach at this point considering the requirements. Thus, attempting to include dynamic gestures along with static ones in the android application and attaining significantly high accuracy for dynamic gesture translation is a goal for future work.

Making the interface for the android application a lot more interactive is also the aim for the near future.

**References**

[1]Byeongkeun Kang , Subarna Tripathi , Truong Q. Nguyen ”Real-time sign language fingerspelling recognition using convolutional neural networks from depth map” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)

[2]Daniels, M. (1994). The effect of sign language on hearing children's language development. *Communication Education*, *43*(4), 291-298.

[3]Han, J., Awad, G., & Sutherland, A. (2009). Automatic skin segmentation and tracking in sign language recognition.

[4]Mahesh, M., Jayaprakash, A., & Geetha, M. (2017, September). Sign language translator for mobile platforms. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1176-1181). IEEE.

[5]Ng, C. W., & Ranganath, S. (2002). Real-time gesture recognition system and application. *Image and Vision computing*, *20*(13-14), 993-1007.

[6] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham

[7] Tan, M., & Le, Q. (2019, May). Efficient Net: Rethinking model scaling for convolutional neural networks. In the International *conference on machine learning* (pp. 6105-6114). PMLR.

[8]Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, *5*(23), 495.

[9]Yang, J., & Xu, Y. (1994). *Hidden markov model for gesture recognition*. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

[10]Ziaie, P., Müller, T., Foster, M. E., & Knoll, A. (2008, March). A naïve Bayes classifier with distance weighting for hand-gesture recognition. In *Computer Society of Iran Computer Conference* (pp. 308-315). Springer, Berlin, Heidelberg.