# A Mini- Project Report

**on**

# “Sign Language Recognition”

Submitted to the

Pune Institute of Computer Technology, Pune

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by


| | |
|---|---|
| Rupesh Mali | 43362 |
| Tanvi Madamwar | 43368 |
| Manish Visave | 43371 |


Under the guidance of

## Prof. R. R. Chhajed


Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.


**2020-2021**

# CERTIFICATE

This is to certify that the project report entitled

**Sign Language Recognition**

**Submitted by**

| | |
|---|---|
| Rupesh Mali | 43362 |
| Tanvi Madamwar | 43368 |
| Manish Visave | 43371 |

is a bonafide work carried out by them under the supervision of Prof. R. R.Chhajed and it is approved for the partial fulfillment of the requirement of **Computer Laboratory -X** for the award of the Degree of Bachelor of Engineering (Information Technology)

**Prof. R. R. Chhajed**
Lab Teacher
Department of Information Technology

**Dr. A. M. Bagade**
Head of Department
Department of Information Technology

Place:
Date:

# ACKNOWLEDGEMENT

We thank everyone who has helped and provided valuable suggestions for successfully creating a wonderful project.

We are very grateful to our guide, Prof. R. R. Chhajed, Head of Department Dr. A. M. Bagade and our principal Dr. P. T. Kulkarni. They have been very supportive and have ensured that all facilities remained available for smooth progress of the project.

We would like to thank our professor and Prof. R. R. Chhajed for providing very valuable and timely suggestions and help.

Rupesh Mali

Tanvi Madamwar

Manish Visave

# ABSTRACT

With a rising demand in disability solutions, sign language recognition has become an important research problem in the field of computer vision. Current sign language detection systems, however, lack the basic characteristics like accessibility and affordability which is necessary for people with speech disability to interact with everyday environment. This project focuses on providing a portable and affordable solution for understanding sign languages by developing an android application. The report provides a summarisation of the basic theory and steps involved in developing this android application that uses gesture recognition to understand letters and digits of Indian sign language. The project uses different image processing tools to separate hand from the rest of the background and then implements pattern recognition techniques for gesture recognition. A comprehensive summarisation of the results obtained from the various tests performed has also been provided to illustrate the efficacy of the application.

# LIST OF FIGURES

# LIST OF TOPICS

# 1. INTRODUCTION

The goal of this project was to build a neural network able to classify which letter of the Indian Sign Language (ISL) alphabet is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written and oral language. Such a translator would greatly lower the barrier for many deaf and mute individuals to be able to better communicate with others in day to day interactions. This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exists at higher rates among the deaf population, especially when they are immersed in a hearing world [1]. Large barriers that profoundly affect life quality stem from the communication disconnect between the deaf and the hearing. Some examples are information deprivation, limitation of social connections, and difficulty integrating in society [2]. Most research implementations for this task have used depth maps generated by depth cameras and high resolution images. The objective of this project was to see if neural networks are able to classify signed ISL letters using simple images of hands taken with a personal device such as a mobile phone camera. This is in alignment with the motivation as this would make a future implementation of a real time ISL-to-oral/written language translator practical in an everyday situation.

The complexity of Indian sign language recognition system increases due to the involvement of both the hands and also the overlapping of the hands. Alphabets and numbers have been recognized successfully. This system can be extended for words and sentences Recognition can be done with PCA (Principal Component analysis)[3]. A multi-class Support Vector Machine (MSVM) can be used for training and recognizing signs of ISL[4].The Euclidean distance transformation is applied on the obtained binary image. Row and column projection is applied on the distance transformed image. For feature extraction, central moments along with HU's moments are used. For classification, neural networks and SVM can be used[5]. The classification of single and double handed Indian sign language recognition using machine learning algorithms can be done with the help of MATLAB with 92-100% accuracy[6].

# 2. SCOPE AND OBJECTIVE

## A. SCOPE

The scope of this project is to build an android application which takes advantage of convolutional neural networks able to classify which letter of the Indian Sign Language (ISL) alphabet and numbers is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written text. Our android application will use this model and determine a sign. There will be a live camera streaming and on button a frame is captured and sign is recognized that is shown in the form of text.

## B. OBJECTIVE

- The objective of this project is to develop an android mobile application that uses sign language recognition to understand Indian sign language.
- The android application should use image processing techniques to distinguish the hand from the background and then identify the fingertips to determine the sign.
- The recognition is limited to only static signs and hence is only used for detecting static Indian sign language alphabets and numbers.
- The android application should understand the gestures and show it in text form.
.
- The project has the following main tasks:
    - Research about different Image Processing techniques.
    - Research about Classification techniques.
    - Sign language recognition model integration with our android application.

# 3. SYSTEM ARCHITECTURE

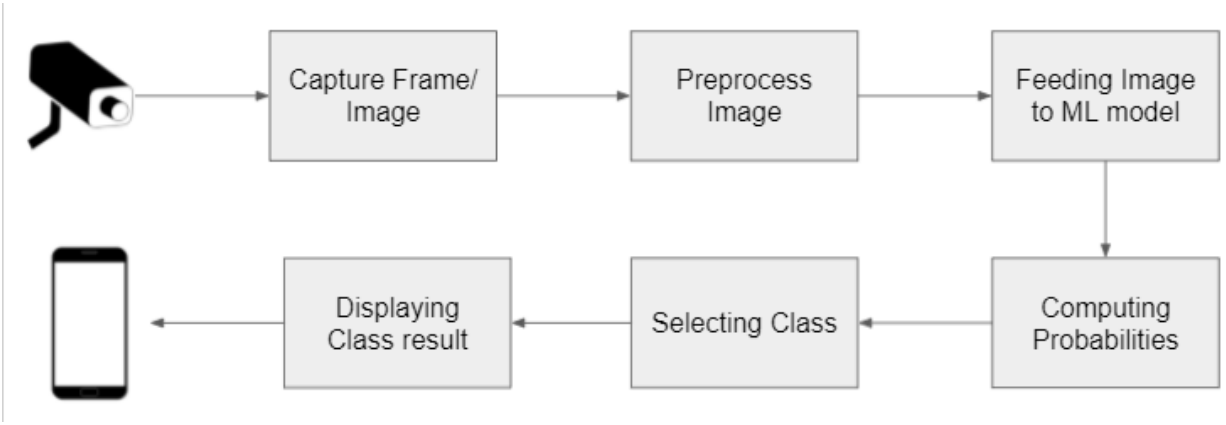


**Figure 1. System Architecture**

1. **Capture Frame/ Image**

   This will be the first step in using the application. As the name suggests, the user needs to capture the image using his smartphone. The image will then be used for further steps in order to get a prediction about the image.

2. **Preprocess Image**

   After capturing the image, it will go through the step of preprocessing. This is the very crucial step as it removes noise from the image. In this step the image is normalized i.e, the pixel values in each channel are transformed into the range 0 to 1. This transformation helps in faster training and inference as the values are between 0 to 1 it makes computations very less.

3. **Feeding Image to the ML model**

   Once the preprocessing step is done, a normalized image is then given to the model as an input. Model we use is the EfficientNet B0 model which is among the best algorithms for image recognition tasks. This model is suitable to run on android mobile as it gives better performance despite being very small.

4. **Computing Probabilities**

   The model gives the output in the form of logits which are basically numeric values. To make it more understanding and to select the appropriate results we need probabilities. Hence these values are transformed into normalized probability using probability mean and probability deviation.

**5. Selecting Class**

After converting logits into probabilities, then we need to select the appropriate class. Basically we have list of classes and probabilities is an array given by model which has same length as the number of classes i.e., every class gets a probability. The most common way of selecting the class is to select the index with the highest probability value among all probabilities. Then we use this index to get the class name from the list of classes.

**6. Displaying Class Result**

We have our result string, we then set this string to the textview in an android application so that the user will know the prediction.

# 4. CODE AND SNAPSHOTS

## CODE

### 1. activity_main.xml

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

    <androidx.constraintlayout.widget.ConstraintLayout
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=".MainActivity">

  <FrameLayout
    android:id="@+id/frameLayout"
    android:layout_width="409dp"
    android:layout_height="500dp"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp"
    tools:ignore="MissingConstraints">

  </FrameLayout>

  <Button
    android:id="@+id/capture"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="61dp"
    android:onClick="captureImage"
    android:text="Capture"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button6"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/frameLayout" />

  <Button
    android:id="@+id/button6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="57dp"
    android:layout_marginEnd="68dp"
    android:text="Clear"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/frameLayout"
```

```
    app:layout_constraintVertical_bias="0.52" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

### 2. MainActivity.java

```
package com.example.slr;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.hardware.Camera;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.FrameLayout;

import com.example.slr.Main2Activity;
import com.example.slr.R;
import com.example.slr.ShowCamera;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

  Camera camera;
  ShowCamera showCamera;
  FrameLayout frameLayout;
  Button capture;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     frameLayout = (FrameLayout) findViewById(R.id.frameLayout);
     capture = (Button) findViewById(R.id.capture);
     //Open Camera
     camera = camera.open();
     showCamera=new ShowCamera(this,camera);
     frameLayout.addView(showCamera);
  }

  Camera.PictureCallback nPictureCallback = new Camera.PictureCallback() {
     @Override
     public void onPictureTaken(byte[] data, Camera camera) {

        Intent in1 = new Intent(MainActivity.this, Main2Activity.class);
        in1.putExtra("image",data);
        startActivity(in1);
     }
  };
```

```
    public void captureImage(View v){
        if(camera!=null){
            camera.takePicture(null,null,nPictureCallback);
        }
    }

    public File getOutputMediaFile() {
        String state = Environment.getExternalStorageState();
        if(!state.equals(Environment.MEDIA_MOUNTED)){
            return null;
        }
        else{
            File folder_gui = new File(Environment.getExternalStorageDirectory() + File.separator+"GUI");
            if(!folder_gui.exists()){
                folder_gui.mkdir();
            }
            File outputFile = new File(folder_gui,"temp.jpg");
            return  outputFile;
        }
    }
}
```

### 3. activity_main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- add Camera Button to open the Camera-->
<!-- add ImageView to display the captured image-->

<Button
    android:id="@+id/button2"
    android:layout_width="100dp"
    android:layout_height="50dp"
    android:layout_marginTop="64dp"
    android:text="Camera"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/image2" />

<ImageView
    android:id="@+id/image2"
    android:layout_width="350dp"
    android:layout_height="450dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView"
```

```
        android:textSize="30dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 4. MainActivity2.java

```
package com.example.slr;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.res.AssetFileDescriptor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.Interpreter;
import org.tensorflow.lite.support.common.FileUtil;
import org.tensorflow.lite.support.common.TensorOperator;
import org.tensorflow.lite.support.common.TensorProcessor;
import org.tensorflow.lite.support.common.ops.NormalizeOp;
import org.tensorflow.lite.support.image.ImageProcessor;
import org.tensorflow.lite.support.image.TensorImage;
import org.tensorflow.lite.support.image.ops.ResizeOp;
import org.tensorflow.lite.support.image.ops.ResizeWithCropOrPadOp;
import org.tensorflow.lite.support.image.ops.Rot90Op;
import org.tensorflow.lite.support.label.TensorLabel;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.graphics.Matrix;
public class Main2Activity extends AppCompatActivity {
```

```
protected Interpreter tflite;
private MappedByteBuffer tfliteModel;
private TensorImage inputImageBuffer;
private  int imageSizeX;
private  int imageSizeY;
private  TensorBuffer outputProbabilityBuffer;
private  TensorProcessor probabilityProcessor;
private static final float IMAGE_MEAN = 0.0f;
private static final float IMAGE_STD = 255.0f;
private static final float PROBABILITY_MEAN = 0.0f;
private static final float PROBABILITY_STD = 1.0f;

Bitmap bitmap;
private List<String> labels;
ImageView imageView;
Uri imageuri;
Button buclassify;
TextView classitext;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    imageView=(ImageView)findViewById(R.id.image2);
    buclassify=(Button)findViewById(R.id.button2);
    classitext=(TextView)findViewById(R.id.textView);
    byte[] byteArray = getIntent().getByteArrayExtra("image");
    bitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);
    bitmap = rotateImage(bitmap,90);
    imageView.setImageBitmap(bitmap);



    try{
        tflite=new Interpreter(loadmodelfile(this));
        int imageTensorIndex = 0;
        int[] imageShape = tflite.getInputTensor(imageTensorIndex).shape(); // {1, height, width, 3}
        imageSizeY = imageShape[1];
        imageSizeX = imageShape[2];
        DataType imageDataType = tflite.getInputTensor(imageTensorIndex).dataType();

        int probabilityTensorIndex = 0;
        int[] probabilityShape =
                tflite.getOutputTensor(probabilityTensorIndex).shape(); // {1, NUM_CLASSES}
        DataType probabilityDataType = tflite.getOutputTensor(probabilityTensorIndex).dataType();

        inputImageBuffer = new TensorImage(imageDataType);
        outputProbabilityBuffer = TensorBuffer.createFixedSize(probabilityShape, probabilityDataType);
        probabilityProcessor = new TensorProcessor.Builder().add(getPostprocessNormalizeOp()).build();

        inputImageBuffer = loadImage(bitmap);

        tflite.run(inputImageBuffer.getBuffer(),outputProbabilityBuffer.getBuffer().rewind());
        showresult();
```

```
        }catch (Exception e) {
            e.printStackTrace();
        }

        buclassify.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            }
        });
    }

    private MappedByteBuffer loadmodelfile(Activity activity) throws IOException {
        AssetFileDescriptor fileDescriptor=activity.getAssets().openFd("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);
    }

    public static Bitmap rotateImage(Bitmap source, float angle) {
        Matrix matrix = new Matrix();
        matrix.postRotate(angle);
        return Bitmap.createBitmap(source, 0, 0, source.getWidth(), source.getHeight(),
                matrix, true);
    }

    private TensorOperator getPreprocessNormalizeOp() {
        return new NormalizeOp(IMAGE_MEAN, IMAGE_STD);
    }
    private TensorOperator getPostprocessNormalizeOp(){
        return new NormalizeOp(PROBABILITY_MEAN, PROBABILITY_STD);
    }

    private TensorImage loadImage(final Bitmap bitmap) {
        // Loads bitmap into a TensorImage.
        inputImageBuffer.load(bitmap);

        // Creates processor for the TensorImage.
        int cropSize = Math.min(bitmap.getWidth(), bitmap.getHeight());
        // TODO(b/143564309): Fuse ops inside ImageProcessor.
        ImageProcessor imageProcessor =
                new ImageProcessor.Builder()
                        .add(new ResizeWithCropOrPadOp(cropSize, cropSize))
                        .add(new ResizeOp(imageSizeX, imageSizeY,
    ResizeOp.ResizeMethod.NEAREST_NEIGHBOR))
                        .add(getPreprocessNormalizeOp())
                        .build();
        return imageProcessor.process(inputImageBuffer);
    }

    private void showresult(){

        try{
            labels = FileUtil.loadLabels(this,"labels.txt");
```

```
        }catch (Exception e){
            e.printStackTrace();
        }
        Map<String, Float> labeledProbability =
                new TensorLabel(labels, probabilityProcessor.process(outputProbabilityBuffer))
                        .getMapWithFloatValue();
        float maxValueInMap =(Collections.max(labeledProbability.values()));

        for (Map.Entry<String, Float> entry : labeledProbability.entrySet()) {
            if (entry.getValue()==maxValueInMap) {
                classitext.setText(entry.getKey());
            }
        }
    }
}
```

## SNAPSHOTS

1. **Home Screen**


**Figure 2 : Home Screen**

**2. Result Screen**

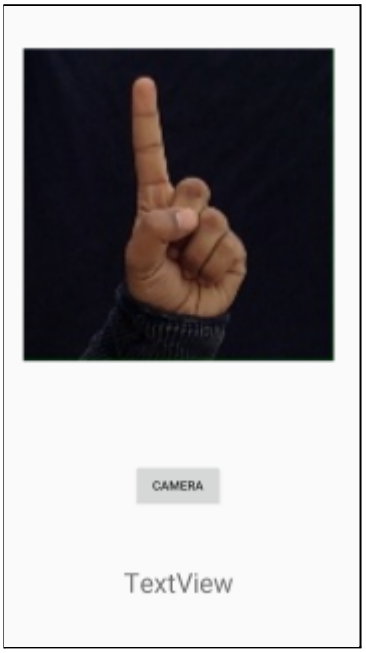


**Figure 3 : Result Screen**

# 5. RESULTS

1) Input and Output for letter V


**Figure 4 : Input V**

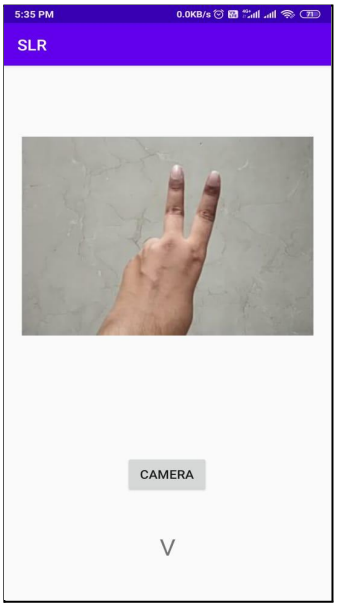


**Figure 5 : Output V**

2) Input and Output for letter L

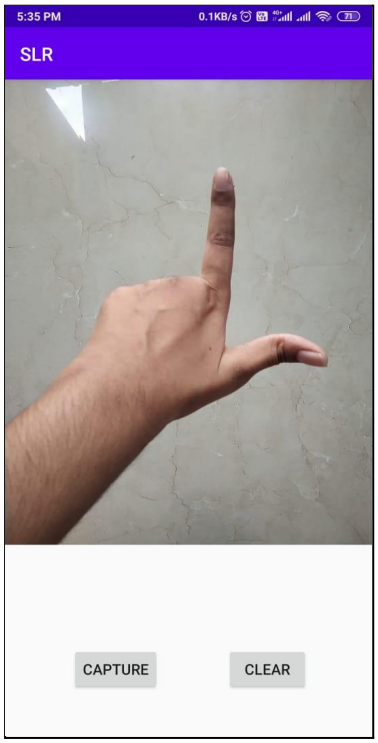


**Figure 6 : Input L**

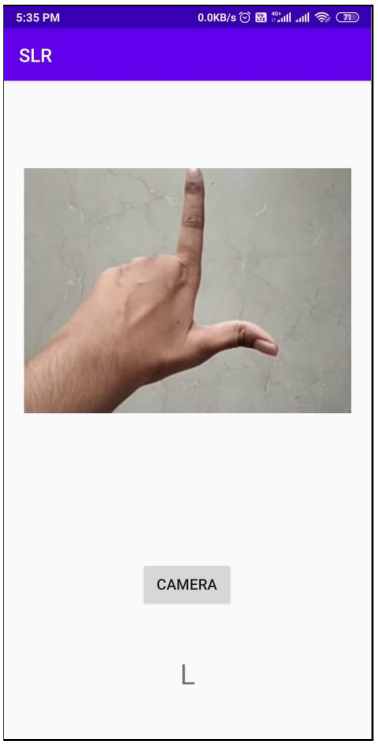


**Figure 7 : Output L**

3) Input and output for digit 5

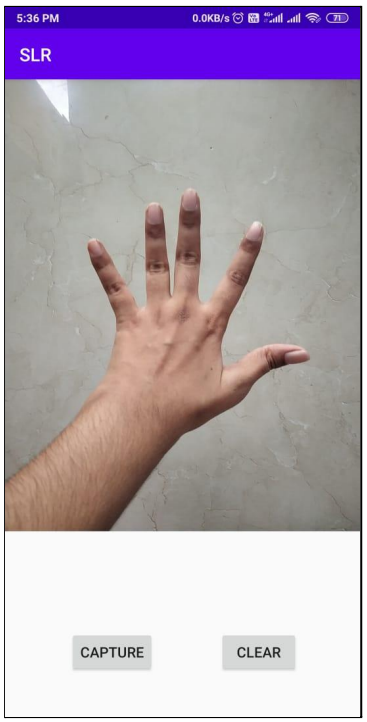


**Figure 8 : Input 5**


**Figure 9 : Output 5**

4) Input and Output for letter O

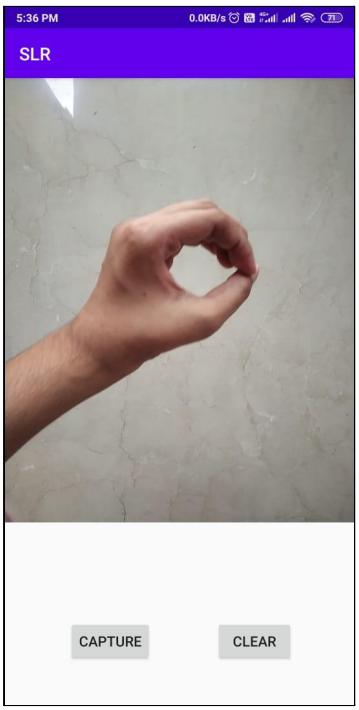


**Figure 10 : Input O**

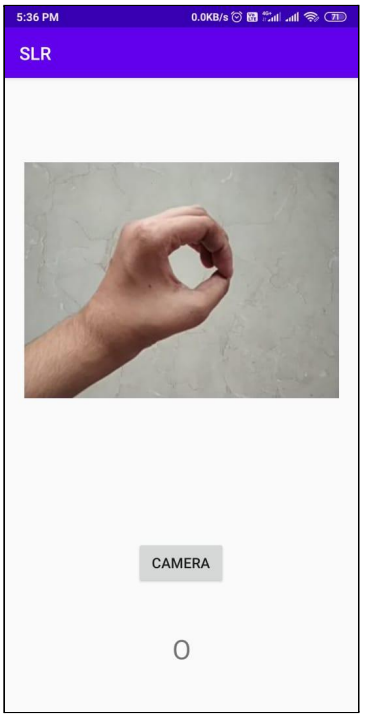


**Figure 11 : Output O**

# 6. CONCLUSION AND FUTURE SCOPE

In this project, we have implemented an automatic sign language gesture recognition system in real-time, using tools learnt in computer vision and machine learning. We learned about how sometimes basic approaches work better than complicated approaches. We have noticed that the recognition could be better if we had proper dataset available with us. The challenge with the available Indian sign language dataset is that we have specific symbols for every character whereas in reality indian sign language has more than one symbol for a single character. Despite of being one of the most accurate models for image recognition EfficientNet hasn't adapted well for real world data, main reason behind it is that dataset available for us had very clean images and it wasn't very noisy and it didn't need any specific transformation except for normalization which is only employed because of speedup in inference and training. But in real time we get an occluded image and there is camera noise in the images. We can improve the performance of the system using the method called skin segmentation. Skin segmentation extracts only the hand part of the body which makes it easier for the model to predict because of less surrounding area and the segmentation is what the model really wants to have as an input. Robustness of the model can also be increased if the appropriate dataset is used while training i.e., if we have a dataset which covers every aspect of indian sign language.

# 7. REFERENCES

[1] Farnaz D. Notash and Elahe Elhamki. “Comparing loneliness, depression and stress in students with hearingimpaired and normal students studying in secondary schools of Tabriz ". In: International Journal of Humanities and Cultural Studies February 2016 Special Issue (2016). issn: 2356-5926.

[2] “The Cognitive, Psychological and Cultural Impact of Communication Barrier on Deaf Adults". In: Journal of Communication Disorders, Deaf Studies Hearing Aids 4 (2 2016). doi: 10.4172/2375-4427.1000164.

[3] D. Deora and N. Bajaj, "Indian sign language recognition," 2012 1st International Conference on Emerging Technology Trends in Electronics, Communication & Networking, 2012, pp. 1-5, doi: 10.1109/ET2ECN.2012.6470093.

[4]K. Dixit and A. S. Jalal, "Automatic Indian Sign Language recognition system," 2013 3rd IEEE International Advance Computing Conference (IACC), 2013, pp. 883-887, doi: 10.1109/IAdCC.2013.6514343.

[5]Rokade, Yogeshwar & Jadav, Prashant. (2017). Indian Sign Language Recognition System. International Journal of Engineering and Technology. 9. 189-196. 10.21817/ijet/2017/v9i3/170903S030.

[6]K. K. Dutta and S. A. S. Bellary, "Machine Learning Techniques for Indian Sign Language Recognition," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017, pp. 333-336, doi: 10.1109/CTCEEC.2017.8454988.