WBSU
Mrinalini Datta Mahavidyapith
Birati, Kolkata 700051
established on 5 March, 1964

Final Project Report

Conversion of Sign Language to text

Under the supervision of
Mr. Sumit Bagchi

Shubhadeep Chakraborty
&
Sourav Dhali

B.Sc CS 3rd year, 6th semester

Reg no.: 1252011400012
&
Reg no.: 1252011400001

2023

Mrinalini Datta Mahavidyapith, West Bengal State University
Birati, Kolkata 700051

CERTIFICATE

This is to certify that the project sign language recognition in python has been prepared by Mr. shubhadeep chakraborty and Mr. sourav dhali, under my supervision, and be accepted in fulfillment of the requirements for the degree of B.Sc honours in Computer science, wbsu.

Mr.Sumit Bagchi , Supervisor
Lecturer, Mrinalini Datta Mahavidyapith
West Bengal State University

Sign Language Recognition

ABSTRACT
Sign Language is mainly used by deaf (hard hearing) and dumbpeople to exchange information between their own community and with other people. It is a language where people use theirhand gestures to communicate as they can't speak or hear. Sign Language Recognition (SLR) deals with recognizing the hand gestures acquisition and continues till text or speech is generated for corresponding hand gestures. Here hand gestures for sign language can be classified as static and dynamic.However, static hand gesture recognition is simpler than dynamic hand gesture recognition, but both recognition is important to the human community.We can use Deep Learning Computer Vision to recognize the hand gestures by building Deep Neural Network architectures (Convolution Neural Network Architectures) where the model will learn to recognize the hand gestures images over an epoch.Once the model Successfully recognizes the gesture the corresponding English text is generated and then text can be converted to speech.This model will be more efficient and hence communicate for the deaf (hard hearing) and dump people will beeasier. In this paper, we will discuss how Sign Language Recognition is done using Deep Learning.

Index words*:*Hand Gestures; Sign Language Recognition; Convolution Neural Networks; Computer Vision; Text to Speech.

1. INTRODUCTION

Deaf (hard hearing) and dumb people use Sign Language (SL) [1] as their primary means toexpress their ideas and thoughtswith their own community and with other people with hand and body gestures.It has its own vocabulary, meaning, and syntax which is different from the spoken language or written language. Spoken language is a language produced by articulate sounds mapped against specific words and grammatical combinations to convey meaningful messages. Sign language uses visual hand and body gestures to convey meaningful messages.There are somewhere between 138 and 300 different types of Sign Language used around globally today. In India, there are only about 250 certified sign language interpreters for a deaf population of around 7 million. This would be a problem to teach sign language to the deaf and dumb people as there isa limitednumber of sign language interpreters exits today.Sign Language Recognition is an attempt to recognize these hand gestures and convert

Fig. 1. Sign Language Hand Gestures

them to the corresponding text or speech.Today Computer Vision and Deep Learning have gained a lot of popularity and many State of the Art (SOTA) models can be built. Using Deep Learning algorithms and Image Processing we can able to classify these hand gestures and able to produce corresponding text.An example of "A" alphabet in sign language notion to English "A" text or speech.

In Deep Learning Convolution Neural Networks (CNN) is the most popular neural network algorithm which is a widely used algorithm for Image/Video tasks. For Convolution Neural Networks (CNN) we have advanced architectures like LeNET-5 [2], and MobileNetV2 [3] where we can use these architectures to achieve the State of the Art (SOTA). We can use all these architectures and combine them using neural network ensemble techniques [4]. By this, we can achieve an almost 100% accurate model which will recognize the hand gestures. This model will be deployed in web frameworks like Django or a standalone application or embedded devices where the hand gestures arerecognized in the live camera and then converting them to text. This system will help deaf and dumb people to communicate easily.

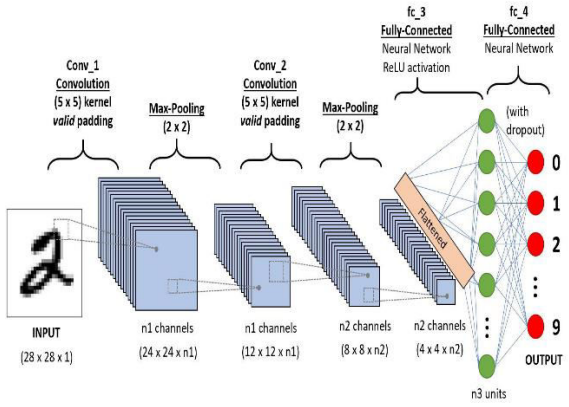


Fig. 2. Convolution Neural Networks

1.1 Motivation

The various advantages of building a Sign Language Recognition system includes:

- ✓ SignLanguage hand gestures totext/speech translation systems or dialog systems which are used in specific public domains such as airports, post offices, or hospitals.
- ✓ Sign Language Recognition (SLR) can help to translate the video to text or speech enables inter-communication between normal and deaf people.

1.2 Problem Statement

Sign language uses lots of gestures so that it looks like movement language which consists of a series of hands and arms motions. For different countries, there are different sign languages and hand gestures.Also, itis noted that some unknown words are translated by simply showing gestures for each alphabet in the word. In addition, sign language also includes specific gestures to each alphabet in the English dictionary and for each number between 0 and 9.

Based on these sign languages are made up of two groups, namely static gesture, and dynamic gesture. The static gesture is used for alphabet and number representation, whereas the dynamic gesture is used for specific concepts. Dynamic also includes words, sentences, etc. The static gesture consists of hand gestures, whereas the latter includes motion of hands, head, or both. Sign language is a visual language and consists of 3 major components, such as finger-spelling, word-level sign vocabulary, and non-manual features. Finger-spelling is used to spell words letter by letter and convey the message whereas the latter is keyword-based. But the design of a sign language translator is quite challenging despite many research efforts during the last few decades. Also, even the same signs have significantly different appearances for different signers and different viewpoints. This work focuses on the creation of a static sign language translator by using a Convolutional Neural Network. We created a lightweight network that can be used with embedded devices/standalone applications/web applications having fewer resources.

1.3 Objectives

The main objectives of this project are to contribute to the field of automatic sign language recognition and translation to text or speech. In our project, we focus on static sign language hand gestures. This work focused on recognizing the hand gestures which includes 26 English alphabets (A-Z) and 10 digits (0-9) using Deep Neural Networks (DNN). We created a convolution neural networks classifier that can classify the hand gestures into English alphabets and digits. We have trained the neural network under different configurations and architectures like LeNet-5 [2], MobileNetV2 [3], and our own architecture. We used the horizontal voting ensemble technique to achieve the maximum accuracy of the model. We have also created a web application using Django Rest Frameworks to test our results from a live camera.

2. LITERATURE REVIEW

2.1Real-time sign language fingerspelling recognition using convolutional neural networks from depth map [5].

This works focuses on static fingerspelling in American Sign Language A method for implementing a sign language to text/voice conversion system without using handheld gloves and sensors, by capturing the gesture continuously and converting them to voice.

In this method, only a few images were captured for recognition. The design of a communication aid for the physically challenged.

2.2Design of a communication aid for physically challenged [6].

The system was developed under the MATLAB environment. It consists of mainly two phases via training phase and the testing phase. In the training phase, the author used feed-forward neural networks. The problem here is MATLAB is not that efficient and also integrating the concurrent attributes as a whole is difficult.

2.3 American Sign Language Interpreter System for Deaf and Dumb Individuals [7].

The discussed procedures could recognize 20 out of 24 static ASL alphabets. The alphabets A, M, N, and S couldn't be recognized due to the occlusion problem. They have used only a limited number of images.

3. IMPLEMENTATION

3.1 Dataset

We have used multiple datasets and trained multiple models to achieve good accuracy.

3.1.1 ASL Alphabet

The data is a collection of images of the alphabet from the American Sign Language, separated into 29 folders that represent the various classes.

The training dataset consists of 87000 images which are 200x200 pixels. There are 29 classes of which 26 are English alphabets A-Z and the rest 3 classes are SPACE, DELETE, and, NOTHING. These 3 classes are very important and helpful in real-time applications.

3.1.2Sign Language Gesture Images Dataset

The dataset consists of 37 different hand sign gestures which include A-Z alphabet gestures, 0-9 number gestures, and also a gesture for space which means how the deaf (hard hearing) and dumb people represent space between two letters or two words while communicating.

Each gesture has 1500 images which are 50x50 pixels, so altogether there are 37 gestures which means there 55,500 images for all gestures. Convolutional Neural Network (CNN) is well suited for this dataset for model training purposes and gesture prediction.

3.2 Data Pre-processing

An image is nothing more than a 2-dimensional array of numbers or pixels which are ranging from 0 to 255.Typically, 0 means black, and 255 means white. Image is defined by mathematical function f(x,y) where 'x' represents horizontal and 'y' represents vertical in a coordinate plane. The value of f(x, y) at any point is giving the pixel value at that point of an image.

Image Pre-processing is the use of algorithms to perform operations on images. It is important to Pre-process the images before sending the images for model training. For example, all the images should have the same size of 200x200 pixels. If not,the model cannot be trained.

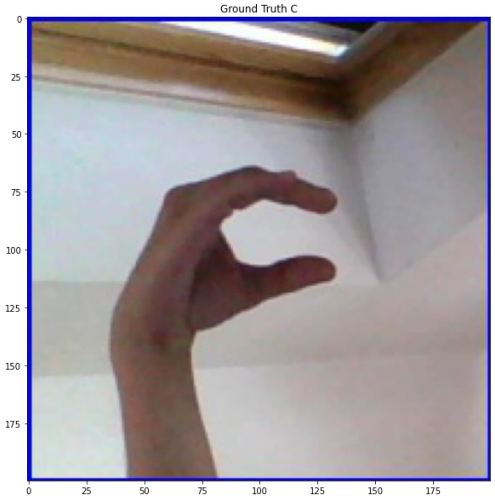


Fig. 3. Sample Image without Pre-processing

Fig. 4. Pre-Processed Image

The steps we have taken for image Pre-processing are:

- ✓ Read Images.
- ✓ Resize or reshape all the images to the same
- ✓ Remove noise.
- ✓ All the image pixels arrays are converted to 0 to 255 by dividing the image array by 255.

3.3 Convolution Neural Networks (CNN)

Computer Vision is a field of Artificial Intelligence that focuses on problems related to images andvideos. CNN combined with Computer vision is capable of performing complex problems.

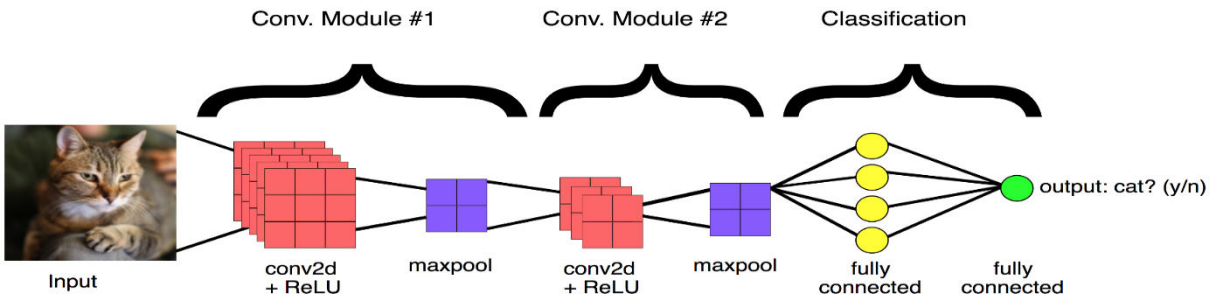


Fig. 5. Working of CNN

The Convolution Neural Networks has two main phases namely feature extraction and classification.

A series of convolution and pooling operations are performed to extract the features of the image.

The size of the output matrixdecreases as we keep on applying the filters.

Size of new matrix = (Size of old matrix — filter size) +1

A fully connected layer in the convolution neural networks will serve as a classifier.

In the last layer, the probability of the class will be predicted.

The main steps involved in convolution neural networks are:

1. Convolution
2. Pooling
3. Flatten
4. Full connection

3.3.1 Convolution

Convolution is nothing but a filter applied to an image to extract the features from it. We will use different filters to extract features like edges, highlighted patterns in an image. The filters will be randomly generated.

What this convolution does is, creates a filter of some size says 3x3 which is the default size. After creating the filter, it starts performing the element-wise multiplication starting from the top left corner of the image to the bottom right of the image.

The obtained results will be extracted feature.

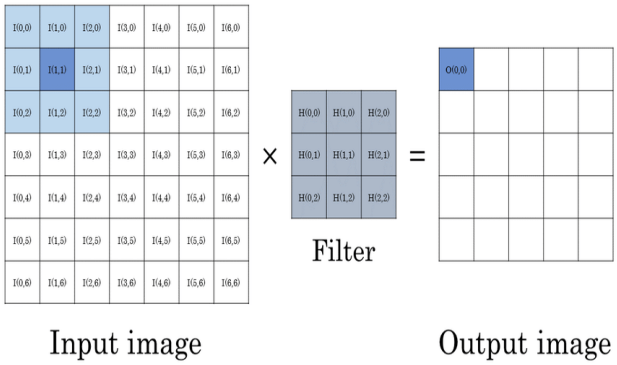


Fig. 6. Convolution

Fig. 6. Feature Extraction

3.3.2 Pooling

After the convolution operation, the pooling layer will be applied. The pooling layer is used to reduce the size of the image. There are two types of pooling:

1. Max Pooling
2. Average Pooling

3.3.2.1 Max pooling

Max pooling is nothing but selecting the maximum pixel value from the matrix.

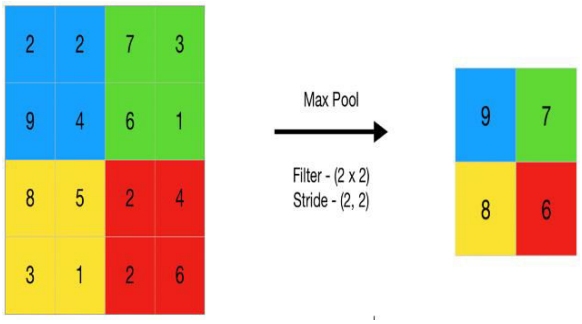


Fig. 7. Max Pooling

This method is helpful to extract the features with high importance or which are highlighted in the image.

3.3.2.2 Average pooling

Unlike Max pooling, the average pooling will take average values of the pixel

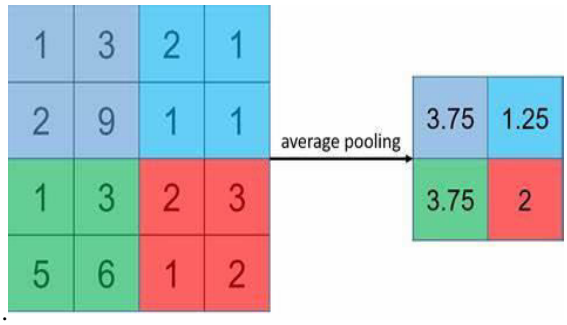


Fig. 8. Average Pooling

In most cases, max pooling is used because its performance is much better than average pooling.

3.3.3 Flatten

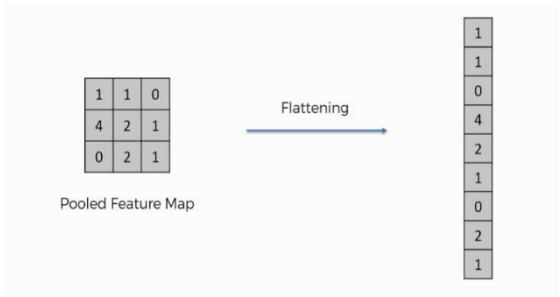
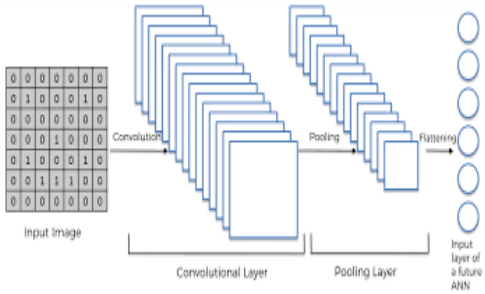


Fig. 9. Flatten

The obtained resultant matrix will be in muti-dimension. Flattening is converting the data into a 1-dimensional array for inputting the layer to the next layer. We flatten the convolution layers to create a single feature vector.



3.3.4 Full Connection

Fig. 10. Full Connection

A fully connected layer is simply a feed-forward neural network. All the operations will be performed and prediction is obtained. Based on the ground truth the loss will be calculated and weights are updated using gradient descent backpropagation algorithm.

3.4 Convolution Neural Network (CNN) Architectures

3.4.1 LeNet-5

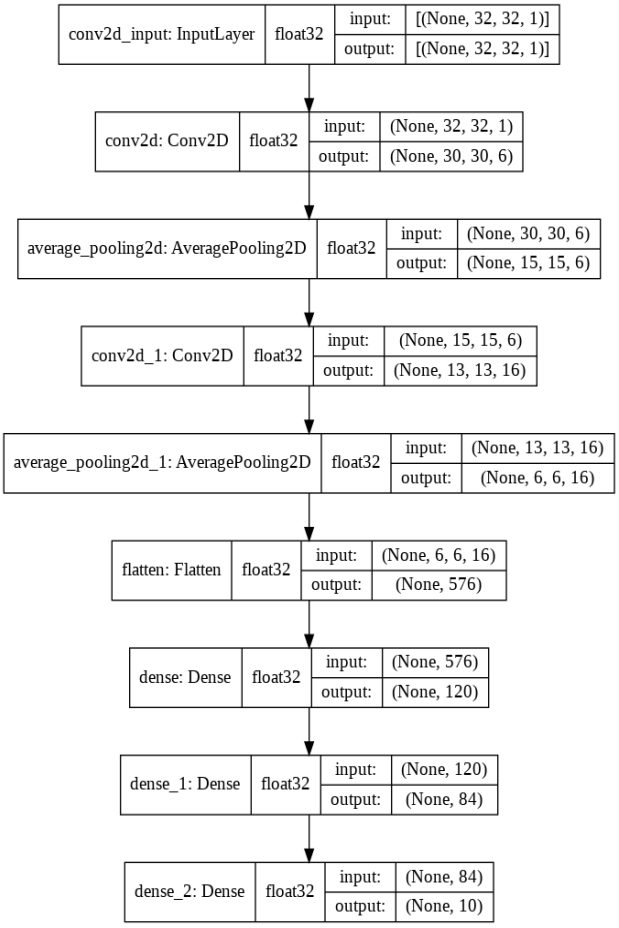


Fig. 11. LeNet-5 Implementation

The LeNet-5 [2] architecture consists of two pairs of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fullyconnected layers, and finally a SoftMax classifier.

3.4.2 MobileNetV2

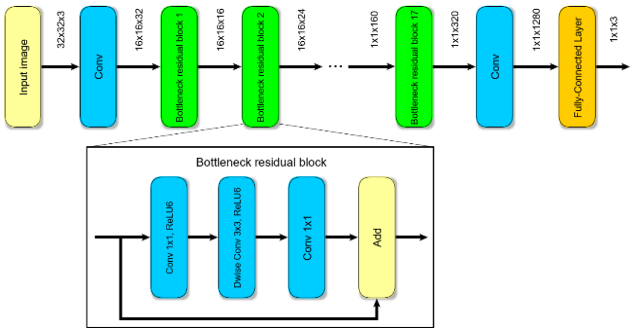


Fig. 12. MobileNetV2 Implementation

MobileNetV2 [3] is a convolutional neural network architecture that performs well on mobile devices.The architecture of MobileNetV2 contains thefully convolution layer with 32 filters, followed by 19 residual bottleneck layers. This network is lightweight and efficient.

3.4.3 Own Architecture

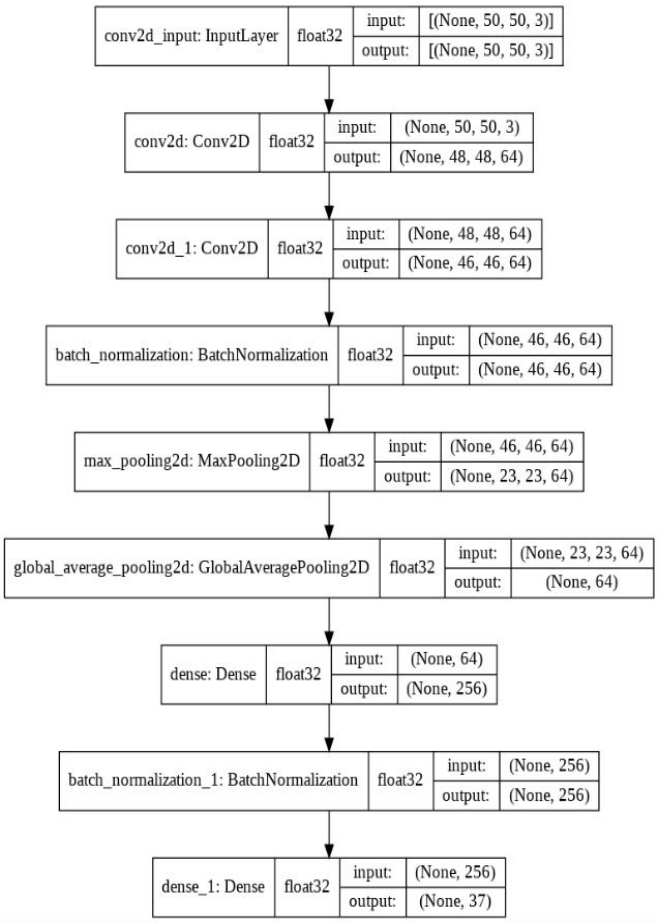


Fig. 13. Own Architecture Implementation

In our own architecture, we have implemented 3 convolution layers followed by batch normalization and max pooling, followed by global average pooling with dense layer and batch normalization, and a final dense layer for classification.

3.5 Proposed Model

We have trained 3 models with 2 different datasets to perform well on unseen datasets. We have trained LeNet-5, MobineNetV2 and, our own architectures. We have not taken the best model out of 3 we have taken all 3 models and made a final model that will perform an ensemble of these 3 models.

3.5.1 Neural Network Ensemble Horizontal Voting

In Machine Learning we have an ensemble technique where we train multiple sub-models and average them. Random Forest algorithm is an example where it uses multiple Decision tree algorithms.Similarly, we can perform ensemble for Neural Networks [4] as well. There are a lot of ensemble techniques for NeuralNetworks like Stacked generalization [8], Ensemble learning via negative correlation [9] and, Probabilistic Modelling with Neural Networks [10] [11]. We have implemented the Horizontal Voting Ensemble method to improve the performance of neural networks.

Horizontal voting is an ensemble technique for neural networks where we train several sub-models and make predictions using these sub-models.For the final predictions, we make predictions from all the sub-models and see which class has got maximum votes.The final prediction will be the class that has the maximum votes. For this, we have used 3 models that arean odd number of sub-models to avoid an even number of votes for two classes in worst cases.

Let model be the set of neural network models being trained on the training set $T(x_i,y_i)$, such that $m \in$ model. Let yhat be the predictions obtained by all the models on the test set $T'(x_i',y_i')$.

Let 'array' be the function for converting lists to arrays

Algorithm 1: Horizontal Voting

Input: models, test set $T'(x_i',y_i')$, and empty yhat list

Output: predictions – final prediction obtained

1. **Step 1:** Obtain the predictions of each model
2. **for** each i in range(x_i)
3. **foreach** m **in** model, **do**
   yhat[i] ←**predict**(x[i])
   Calculate highest number of votes for ith test data and append
   yhat[i] ← highest voted class
   endfor
4. **end for**

5. **Step 2:** Convert list into an array
6. yhat←**array**(yhat)
7. **Step 3: return** yhat

For MobileNetV2 model we have takenAdam optimizer with learning rate = 0.001,eta_1=0.9, beta_2=0.999, and epsilon=1e-07

For all the models while training we have used ReduceLROnPlateaucallback with factor = 0.2, patience = 2, min_lr = 0.001.

By using this horizontal ensemble technique, we have achieved 99.8% accuracy.

We have deployed all the models in Django Web Frameworks and built a simple frontend to accept the image from users and send the response.

We have also built an API that will pop up the live camera and detects the hand gestures and then converts them to the corresponding English alphabets.

4. EXPERIMENTAL RESULTS

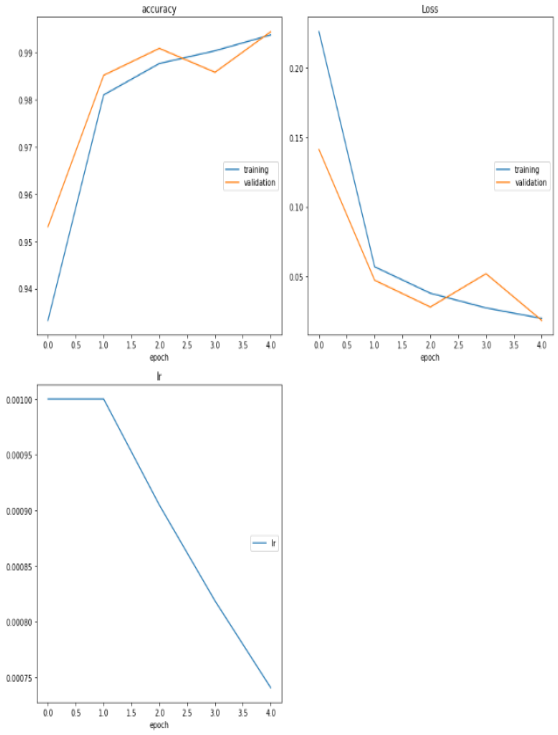


Fig. 14. Traininggraphs

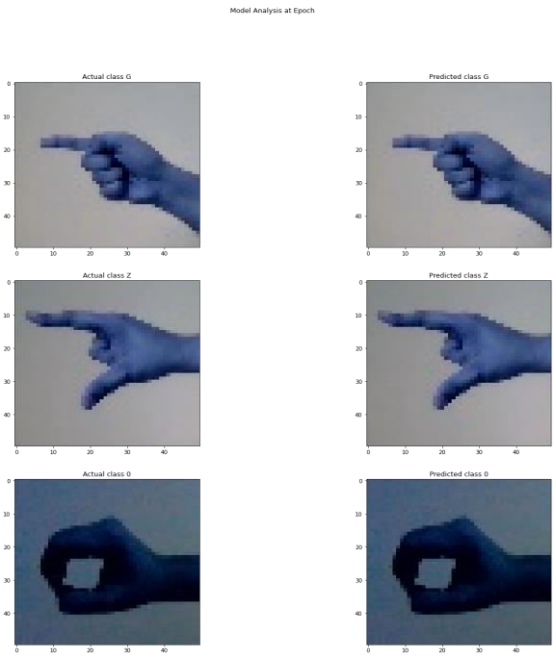


Fig.15. Model Prediction during training

Models	**Accuracy**
MobileNetV2	98.9%
LeNet-5	97%
Own Model	98%
Ensemble	**99.8%**

Table. 1. Performance Results

We have trained all the models for around 10-15 epochs with a batch size of 32.

All the models performed well on the test cases. After applying the horizontal voting ensemble technique for these 3 models, we have achieved almost 100% accuracy.

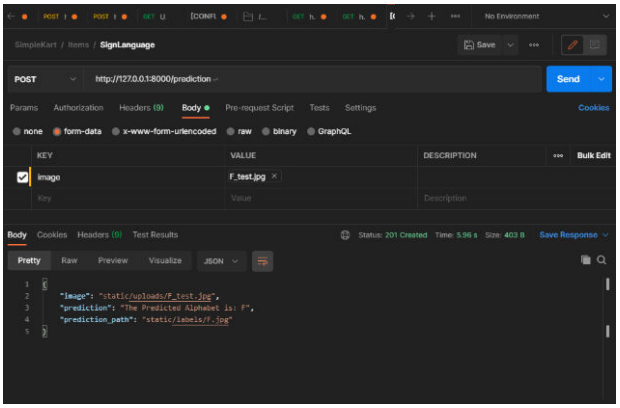


Fig. 16. JSON Response from the Application

We have used Django Rest Frameworks as a backend for our project. This is the sample JSON response that is sent to the frontend when a user inputs an Image. We have used OpenCV to test our results in the live camera. This is a sample result on a live camera

Fig. 17. Model Prediction on Live Camera Prediction: "W"

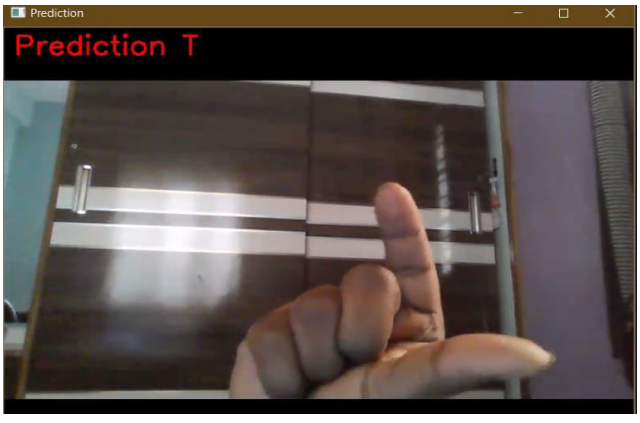


Fig. 18. Model Prediction on Live Camera Prediction: "T

Fig. 19. Model Prediction on Live Camera Prediction: "C"

Fig. 20. Model Prediction on Live Camera Prediction: "M"

5. CONCLUSION

In conclusion, we were successfully able to develop a practical and meaningful system that can able to understand sign language and translate that to the corresponding text. There are still many shortages of our system like this system can detect 0-9 digits and A-Z alphabets hand gestures but doesn't cover body gestures and other dynamic gestures. We are sure and it can be improved and optimized in the future.

Code Implementation

Required Libraries:

1. opencv-python ---------- #For camera
2. tensorflow ---------- #For classifier
3. mediapipe -------- #For numpy and more
4. cvzone - - - - - - - - #For hand tracking

Starting the Webcam:

```
import cv2
cap=cv2.VideoCapture(0)
while True:
        success,img=cap.read()
        cv2.imshow("^-^",imgOutput)
        cv2.waitKey(1)
```

If there is no hand on the camera then the display not showing alphabet signs

Testing Code:

```
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
classifier=Classifier('model/keras_model.h5','model/labels.txt')
labels=['A','B','C','D','E','F','G','I','K','L','O','P',
                    'Q','W','X','Y','Like','Dislike','Rock','Thank you']
```

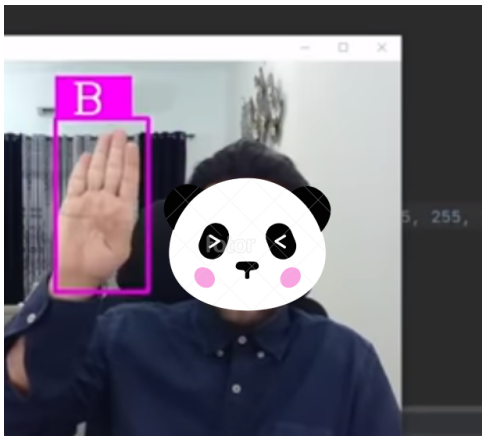
Then we take all alphabet and scan the hand sign, make a secparate image of hand sign and the model classified the image scan and output the alphabet.

Our full Code:

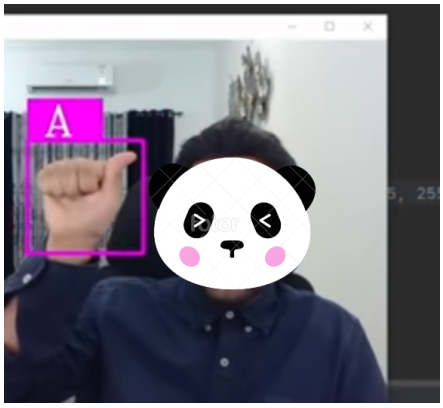
```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
cap=cv2.VideoCapture(0)
detector=HandDetector(maxHands=1)
classifier=Classifier('model/keras_model.h5','model/labels.txt')
offset=20
imgSize=300
labels=['A','B','C','D','E','F','G','I','K','L','O','P',
                                'Q','W','X','Y','Like','Dislike','Rock','Thank you']
while True:
    success,img=cap.read()
    imgOutput=img.copy()
    hands,img=detector.findHands(img)
    if hands:
        hand=hands[0]
        x,y,w,h=hand['bbox']
        imgWhite=np.ones((imgSize,imgSize,3),np.uint8)
        imgCrop=img[y-offset:y+h+offset,x-offset:x+w+offset]
        aspectRatio=h/w
        if aspectRatio>1:
            k=imgSize/h
            wCal=math.ceil(k*w)
            imgResize=cv2.resize(imgCrop,(wCal,imgSize))
            imgResizeShape=imgResize.shape
            wGap=math.ceil((imgSize-wCal)/2)
            imgWhite[:,wGap:wCal+wGap]=imgResize
            prediction,index=classifier.getPrediction(imgWhite,draw=False)
        else:
            k=imgSize/w
            hCal=math.ceil(k*h)
            imgResize=cv2.resize(imgCrop,(imgSize,hCal))
            imgResizeShape=imgResize.shape
            hGap=math.ceil((imgSize-hCal)/2)
            imgWhite[hGap:hCal+hGap,:]=imgResize
            prediction,index=classifier.getPrediction(imgWhite,draw=False)
        cv2.rectangle(imgOutput,(x-offset,y-offset-50),
                      (x-offset+90,y-offset-50+50),(255,0,255),cv2.FILLED)
        cv2.putText(imgOutput,labels[index],(x,y-26),
                    cv2.FONT_HERSHEY_COMPLEX,1.7,(255,255,255),2)
        cv2.rectangle(imgOutput,(x-offset,y-offset),
                                    (x+w+offset,y+h+offset),(255,0,255),4)
    cv2.imshow("^-^",imgOutput)
    cv2.waitKey(1)
```

outputs:

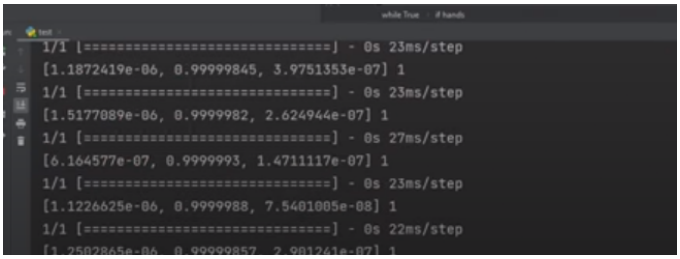
A-pic



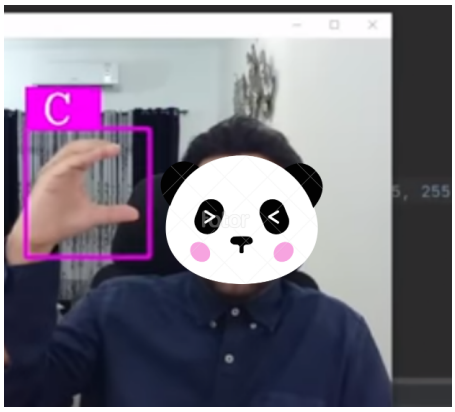
B-pic



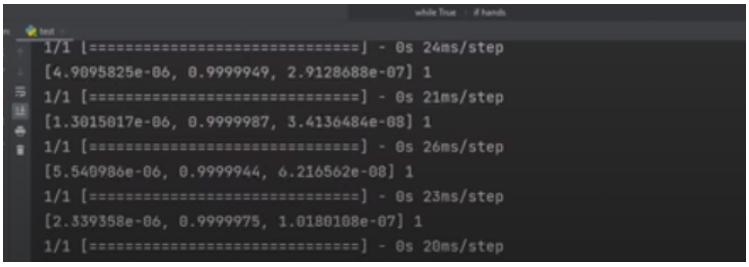
terminal



C-pic



terminal



REFERENCES

[1] Brill R. 1986. The Conference of Educational Administrators Serving the Deaf: A History. Washington, DC: Gallaudet University Press.

[2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[4] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.

[5] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-9326-7_1.