Bangladesh University of Engineering and Technology
Department of Electrical & Electronic Engineering

EEE 312 Project Report

Project Title:

American Sign Language Recognition and Live Detection

Submitted To

Dr. S. M. Mahbubur Rahman
Professor, Department of EEE, BUET
Shahed Ahmed
Lecturer, Department of EEE, BUET

Submitted By:

Group no: 5

Name	Student ID
Shoumik Debnath	1706050
Sariha Noor Azad	1706051
Nazmus Salehin	1706052
Saleh Ahmed Khan	1706053

Abstract:

Communication between deaf and normal people has been one of the challenges in our society as mass people aren't familiar with ASL signs. In this age of globalization, it's our responsibility to help out our deaf community. This project presents a system that recognizes the letter hand gesture of American Sign Language. This system is expected to recognize the gestures of deaf people and convert them into letter for normal people. The process involves Machine Learning, Convolutional Neural Network (CNN) for training and to classify image and Media-Pipe python library for hand tracking. The accuracy of our results shows the effectiveness of this project.

Keywords:

American Sign Language (ASL), Machine Learning, Gesture Recognition, Convolutional Neural Network (CNN), Media-Pipe Python Library, Live detection, Google Colab.

Introduction:

American Sign Language (ASL) is a complete and generalized visual language that is expressed by facial expression as well as movement and motion with hands. ASL serves as the predominant sign language for deaf-mute communities. ASL uses an array for motions, gestures, body languages and hand signs for expressing words, letters and sentences.
Detecting the sign language can be done in two levels, i.e., word level (based on gestures and body language) and letter level (based on hand signals).

Figure (I) Gesture for the word "Hello" (word level) (II) Sign for the letter "H" (letter level)

In our project is based on letter level detection. Because, it is much simpler than word level detection which requires gesture/motion detection and is considerably more complex.

We have used a MATLAB code to take pictures of our own hand from live video. We have added this to our dataset. Then we have trained the dataset with a convolutional neural network model in Google Colab. We have saved our final code in google drive with public access.

Literature Review:

American Sign Language (ASL) is the predominant sign language of deaf communities in the United States and Canada. It was invented by Thomas Hopkins Gallaudet in 1817. In the last two centuries the communication between deaf individuals has been done through interpreters or writings. There have been explorations done in this field to recognize signs in videos and images using several methods and algorithms. However, there has been some work of live ASL detection has been done in the past [2]

Objectives of our Project:

1. Our Convolutional Neural Network is trained with the objective to be able to distinguish between 39 cases of letter level ASL. From the letters A to Z as well as numbers for 0 to 9 and three special cases for a blank (no hand in the picture), space and delete.
2. The input of our program is the portrait/silhouette of a person holding up their hands. In order to pass their hand gesture through our CNN, the input picture must be localized only to the person's hand. So, our program must be able to track, detect and crop the region around the hands.
3. Our program is created with the mindset to make the program/software available and accessible to everyone. Therefore, the program and its essential files must be available in an online public server and anyone should be able to access and run the program without any installation.

Methodology:

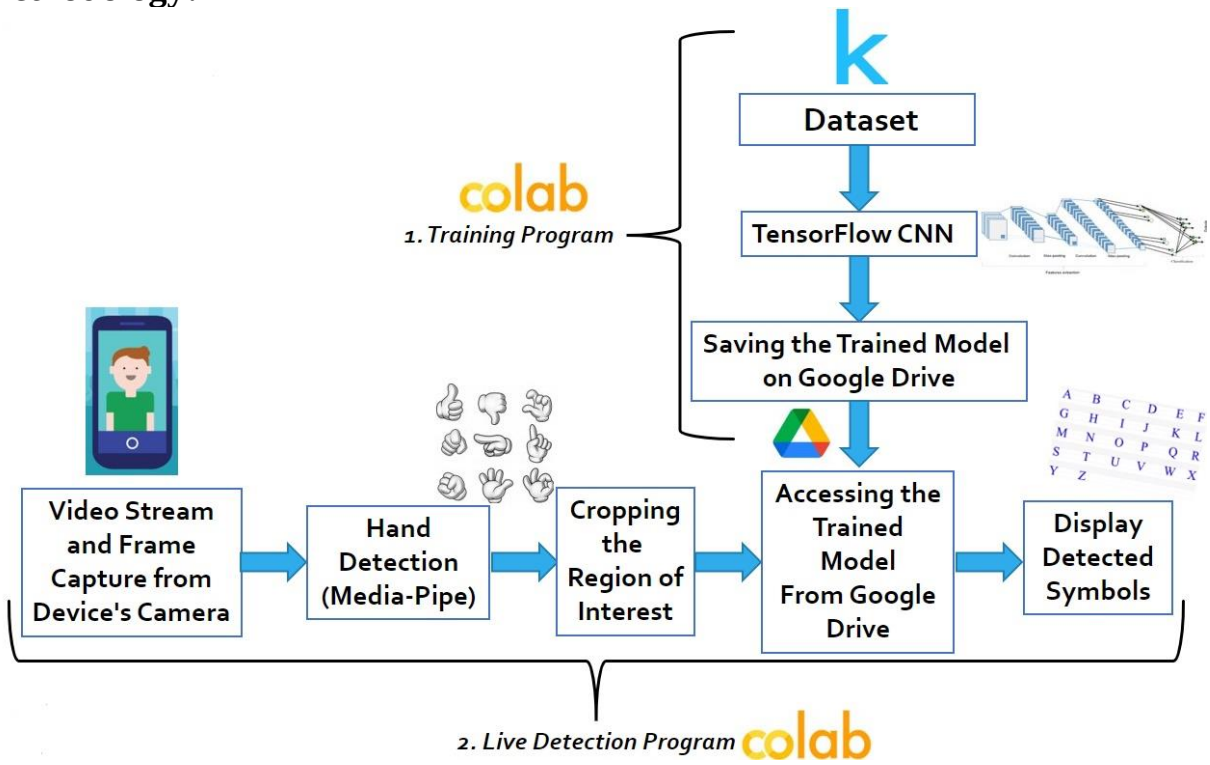

Figure: Workflow of our Program

Phase 1: Training the Model

(i) Dataset Creation:

A proper dataset must be built containing multiple examples of all 39 cases of symbols. Also, our program should be trained to satisfactorily detect live images. So, a combination of pre-built datasets from Kaggle (https://www.kaggle.com)[5][6], as well as live data collected by the team members is used as the final dataset.

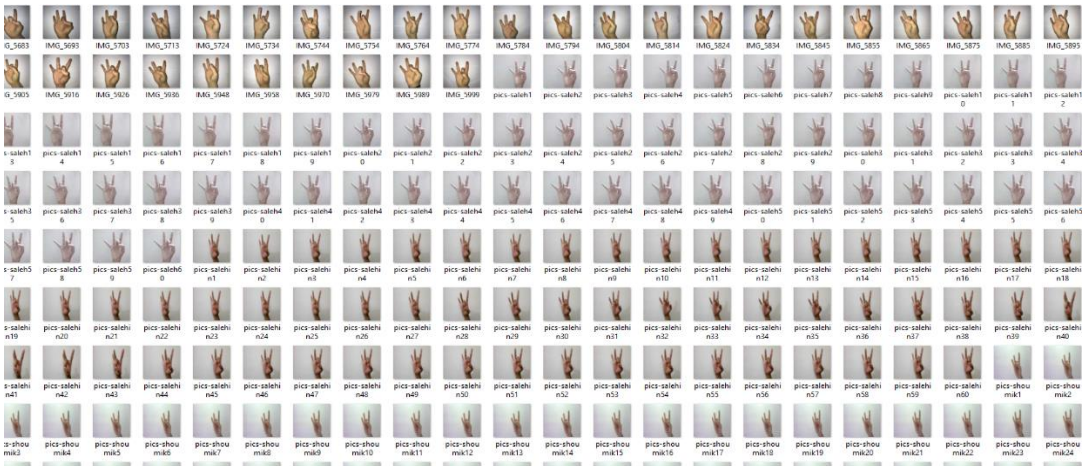

Figure: Training Datasets

Dataset Details:

Total Cases: 39
Images per Case: 380 (200 from Kaggle, 180 live pictues collected by team members)
Image Size: 100x100x3 pixels (RGB images)
Total Images: 14820
Data Allocation: 80% on training data, 10% on validation set, 10% on test set.

The live data were collected by team members using a MATLAB which can collect a burst of images which in a few seconds and save them in the appropriate directory for convenience.

The final dataset matrix and labels matrix were also created using a MATLAB program which saved the entire dataset in two .mat files, which are then transferred to Google Colab.

(ii) Building the Convolutional Neural Network:

Among the currently available deep learning networks, CNN is one of the best networks to use when it comes to processing images and learning patterns. Therefore, we have opted for a multi layered CNN with a couple of fully connected layers for our model. Our model is built using TensorFlow Libraries in Google Colab's deep learning and research platform.
Details about our Convolutional Neural Network is shown in this picture.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 264)               1081608
_________________________________________________________________
dense_1 (Dense)              (None, 39)                10335
=================================================================
Total params: 1,185,191
Trainable params: 1,185,191
Non-trainable params: 0
_________________________________________________________________
```

Figure: Convolutional Neural Network Details

(iii) Training and Saving the Model:

Finally, the model has been trained using the datasets and saved as a TensorFlow model of Google Drive's Public Access, so that Phase 2 of our model can easily access it from any device.

```
Epoch 1/7
325/325 [==============================] - 170s 519ms/step - loss: 2.2157 - accuracy: 0.4399 - val_loss: 0.9797 - val_accuracy: 0.7860
Epoch 2/7
325/325 [==============================] - 168s 517ms/step - loss: 0.5434 - accuracy: 0.8947 - val_loss: 0.3627 - val_accuracy: 0.9176
Epoch 3/7
325/325 [==============================] - 168s 516ms/step - loss: 0.2013 - accuracy: 0.9703 - val_loss: 0.1865 - val_accuracy: 0.9646
Epoch 4/7
325/325 [==============================] - 167s 514ms/step - loss: 0.0825 - accuracy: 0.9929 - val_loss: 0.1049 - val_accuracy: 0.9792
Epoch 5/7
325/325 [==============================] - 168s 516ms/step - loss: 0.0394 - accuracy: 0.9981 - val_loss: 0.0789 - val_accuracy: 0.9838
Epoch 6/7
325/325 [==============================] - 168s 515ms/step - loss: 0.0193 - accuracy: 0.9995 - val_loss: 0.0557 - val_accuracy: 0.9915
Epoch 7/7
325/325 [==============================] - 167s 515ms/step - loss: 0.0114 - accuracy: 0.9998 - val_loss: 0.0510 - val_accuracy: 0.9885
41/41 [==============================] - 5s 129ms/step - loss: 0.0692 - accuracy: 0.9831
[0.06918662786483765, 0.9830769300460815]
```

Figure: Training of our model

Model Fitting and Evaluation Parameters

Optimizer: *Adam Optimization Function*
Loss Function: *Sparse Categorical Cross Entropy*
Evaluation Metric: *Accuracy*

Total Epochs: *7*
Total Training Time: *19 minutes, 36 seconds*
Training Set Accuracy: *99.98 %*
Validation Set Accuracy: *98.85 %*
Testing Set Accuracy: *98.30 %*

Phase 2: Live Detection Program

Fig. Live Detection

(i) Video Stream and Frame Capture using Device's Camera:

In order to detect live gestures, first our program needs to stream video feed for the device's camera or webcam, then capture individual frames and process those frames. In order to stream video from Google Colab provides a prebuilt code, that allows the user to link their local device's camera to Google Colab's computer on the cloud.
However, frame detection is slow and occurs at 1 frame per second, due to streaming device and the main processor being on two different physical locations.

(ii) Hand Detection and Cropping:

After capturing a person's portrait/silhouette, their location of their wrist is tracked using Python's Media-Pipe Hand Detection Library (https://pypi.org/project/mediapipe/). After extracting the location, a red bounding box is placed around the region of interest and the image is cropped and stored in a variable which is then used as input for our previously trained network.

(iii) *Letter Recognition and Displaying Output:*

This program accesses the previously saved training network from Google Drive, then uses it to detect the sign language captured by the device's camera.
If a match is found, the program outputs the corresponding label.
If no match is found, the system will output "cannot detect" message.
If there is no hand in the frame, the system will also output "cannot detect" message.

Results and Discussions:

Firstly, our Convolutional Neural Network shows promising results on our datasets with a very good accuracy on the training data and satisfactory accuracy on testing data which includes live examples.
Therefore, our model is a suitable choice for a Sign Language Detection Program.

Afterwards, we move on to live testing of our data. The system was tested on four users (all our group-mates) in our home under various lighting conditions and background. We would like to mention here that we have also tested on people who have zero knowledge of our project.

Also, the program has been shared with people outside the group to see whether they can access the GitHub repository and run the program in Google Colab to ensure anyone can utilize our project for humanitarian purposes when the need arises.

Some sample live detection pictures and results for all 39 cases are shown below:

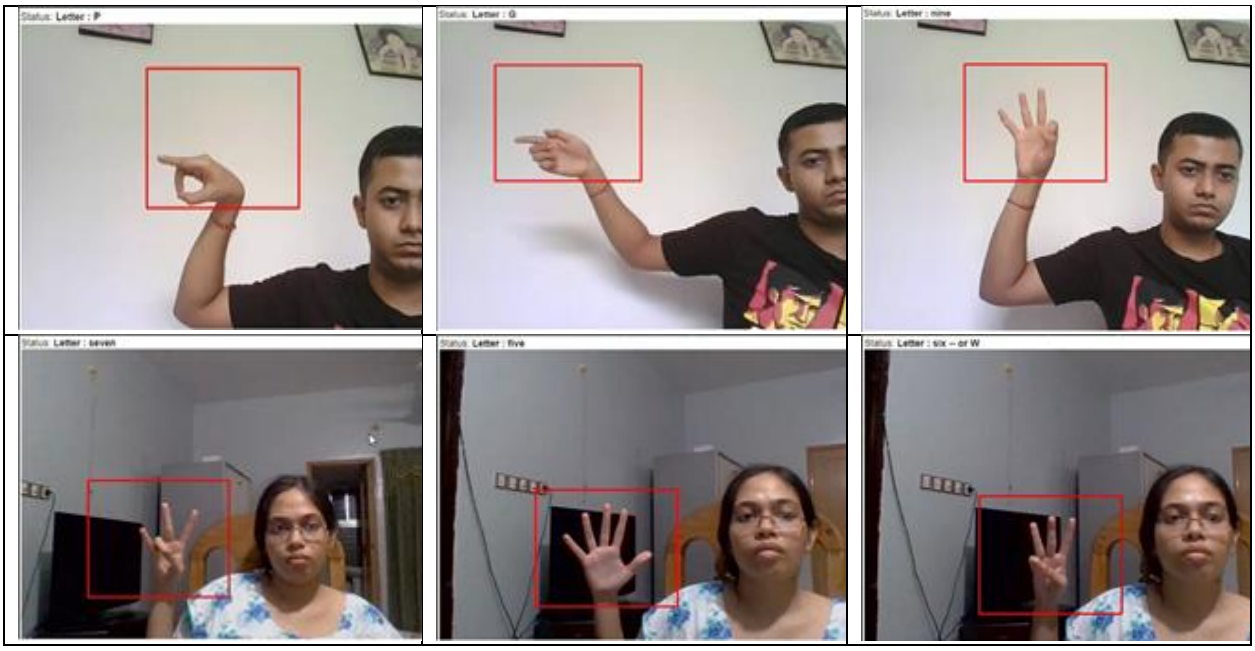

Fig. Live Detection Pictures

As we can see from the above figures, our system can detect accurate signs in almost all the cases. This implies that our system can be used in practical cases for helping the disabled.

Detection Summary Table

Detection Performance	Cases (letters and digits)	Number of Cases
Excellent Detection and Tracking	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, K, L, O, P, S, V, W, Y, space, blank	29
Average Detection	J, N, R	3
Oscillating/Unstable Detection	M, X, U, Z	4
Cannot Detect Satisfactorily	Q, T, del	3
	Total Tested Cases	39

Reasons for Poor Detection Performance:

1. *Overlapping/ Very Close resemblance to another letter/digit*

   Notable Issues: (digit 2, V and K), (M, N), (6 and W), (R and U), (T and X) have similar patterns, and therefore hard to detect on a live testing.

2. *Hand detection software unable to track the gesture accurately*

   Notable Issues: (Del and Q) have their gestures set up in such a manner which is difficult for the Media-Pipe library to detect the user's wrist. And thus, it is difficult to detect these letters accurately

3. *Lighting, Background, Video Quality and Noise*

   Live testing takes place in a completely different environment than dataset images, which also affects performance and tracking.

GitHub Repository and Public Access:

As mentioned before, our program must be easily accessible to anyone who want to communicate with a sign-user. Therefore, our program has been uploaded to GitHub for ease of access to everyone. Also, all the datasets, trained models and necessary codes are also available on GitHub. So, anyone wishing to use our model, our work on their own projects in the future will have access to our materials and research in this humanitarian endeavor.

GitHub Link to the Project:
https://github.com/CurtisTelvanni/Live-American-Sign-Language-Recognition-EEE-312-Project

Future scope:

In our project, we have tried to detect letters and digits. The accuracy level of the detection was satisfactory. Despite our success, we do not wish to stop here as the whole project seems promising to us. We have already mentioned that the sole purpose of this project is to make communication easier with deaf mute people among us by helping them express their feelings, and enabling everyone to understand that. Keeping this in our minds we want to take this project some steps further. We wish to develop a system that can not only identify digits and letters but also words, phrases, sentences. We may even try to make this text to speech so that the system can read aloud the identified sentences, words. Also, we can train different models in the future to suit different languages (For example: Bangla Sign Language). This will help us to reach a broader level of audiences. And as we have already made this project open to all, we believe these improvisations will make this project far more practical. And finally add the reverse process (Voice to Sign) in our program to ease communication between a sign and a non-sign user.

Drawbacks:

In spite of our hard work and a higher level of accuracy, we still have few drawbacks in the project. The main drawback is that our live detection is not robust and it is also slow. It may take a few seconds to detect a letter or digit. This happens because of the fact that we are using Google Colab which is an online platform. We can make the detection fast by using the program offline in a native computer. But our sole purpose is to reach out to the disabled people who actually need help, we had no other option other than using the online based platform.

Conclusion:

This project was aimed at developing a live ASL letter detection system that narrows the communication barrier between deaf and normal people. Our project detects letter in the shortest time with a remarkable accuracy. This project is also implementable in any smartphone device that contains a camera and a Google account. Here, it has been shown that how CNN can be used to detect ASL signs.

Acknowledgements:

No word is enough to express gratitude to our teachers for guiding us through the project. We pay our utmost respect to Dr. S. M. Mahbubur Rahman, Professor, Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, and Shahed Ahmed, Lecturer, Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, for their valuable suggestion and supervision in this project.

References:

[1] https://www.researchgate.net/publication/332372103_Fine-tuning_a_pre-trained_Convolutional_Neural_Network_Model_to_translate_American_Sign_Language_in_Real-time/figures?lo=1
[2] Teena Varma, Ricketa Baptista, Daksha Chithirai Pandi, Ryland Coutinho.
“Sign Language Detection using Image Processing and Deep Learning”
[3]
https://www.researchgate.net/publication/326972551_American_Sign_Language_Recognition_System_An_Optimal_Approach
[4] GitHub link to the Project and all codes
https://github.com/CurtisTelvanni/Live-American-Sign-Language-Recognition-EEE-312-Project
[5] ASL letters dataset from Kaggle
https://www.kaggle.com/grassknoted/asl-alphabet
[6] ASL numbers dataset from Kaggle
https://www.kaggle.com/ardamavi/sign-language-digits-dataset
[7] Drive Link to the dataset used for the project
https://drive.google.com/drive/folders/1yqFp-_nITNoepTeGPSoMOzGA1lB1E6GC?usp=sharing
[8] Drive Link to the trained model
https://drive.google.com/drive/folders/1-yB_O0AkgzYt4uWSWC7NqIaKQzqNuPTz?usp=sharing