Sign Language Recognition System

Project Report

Submitted by

Mahesh Randale	**111708050**
Rushikesh Sabale	**111708052**
Sahil Jadhav	**111708053**

in partial fulfilment for the award of the degree of

B.Tech (Information Technology)

Under the guidance of

Prof. Shirish Gosavi

College of Engineering, Pune

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE-5

May, 2021

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

Certified that the project titled, “Sign Language Recognition System” has been successfully completed by

Mahesh M. Randale	**111708050**
Rushikesh R. Sabale	**111708052**
Sahil N. Jadhav	**111708053**

and is approved for the partial fulfillment of the requirements for the degree of “B.Tech. Information Technology”.

SIGNATURE
Prof. Shirish Gosavi
Project Guide
Department of Computer Engineering
and Information Technology,
College of Engineering Pune,
Shivajinagar, Pune - 5.

SIGNATURE
Dr.(Mrs.) V.Z.Attar
Head
Department of Computer Engineering
and Information Technology,
College of Engineering Pune,
Shivajinagar, Pune - 5.

Abstract

A large number of deaf and mute people are present around the world and communicating with them is a bit difficult at times; because not everyone can understand Sign language(a system of communication using visual gestures and signs). In addition, there is a lack of official sign language interpreters. In India, the official number of approved sign language interpreters is only 250[1] . This makes communication with deaf and mute people very difficult. The majority of deaf and dumb teaching methods involve accommodating them to people who do not have disabilities - while discouraging the use of sign language. There is a need to encourage the use of sign language. People communicate with each other in sign language by using hand and finger gestures. The language serves its purpose by bridging the gap between the deaf-mute and speaking communities. Sign language identification is a challenging area in the field of computer vision, and recent developments have made some progress but not been able to build an efficient system with many challenges still to be solved. In this project, we propose an optimal recognition engine whose main objective is to translate static American Sign language alphabets, numbers, and words into human and machine understandable English script and the other way around. We propose a machine learning based approach for American Sign Language identification using Neural Networks.

Contents

List of Tables

List of Figures

List of Symbols

Chapter 1

Introduction

The majority of the world's deaf community uses American Sign Language(ASL), a visual gesture language. The use of a sign language recognition system is also quite important and necessary in a country like India. ASL is a globally recognised standard for sign language, but only a small number of people understand it, limiting user's ability to converse in real-life scenarios. We propose an ASL recognition system in this project to address this issue and thus make communication between the speaking and non-speaking communities much easier and simpler. The proposed system employs a series of pre-processing steps to convert a gesture image into boundary highlighted images that are then fed into a machine learning algorithm for identification. Existing solutions necessitate the use of external devices to capture finger movements, such as motion sensing gloves or the Microsoft Kinect. As a result, the feasibility and accessibility are reduced. Furthermore, they employ a highly complex system of neural networks, which is computationally expensive. The systems rely heavily on hardware components in their operation, which raises the system cost and makes it difficult to use for ordinary people.

1.1 Nature and Scope of Project

With the backing of advanced technology, as personal interpreters for deaf and mute people are limited, the need of a computer system to carry out the task of sign language recognition is important. With advancements in science and technology, one can consider developing a highly efficient framework that interprets gestures into human or machine understandable text, making it easier for common people to understand and converse with the hearing impaired society.

1.2 Motivation

With many constraints, factors and limitations, Sign Language Recognition(SLR) is a challenging and motivating task. SLR is an important task because of the impact it has on society in terms of bridging the communication gap between the speaking and non-speaking communities. Sign Language Recognition system is difficult to implement because of the variations in hand gestures, facial expressions, body movements and many other variations and constraints. The social impact, the limitations in current implementations, cost constraints etc. were the motivating factors for developing this system.

1.3 Principal objectives

- Acquisition of gesture image
- Pre-processing the input image
- Translation from treated image to text
- Speech to Sign conversion for effortless two way communication.

Chapter 2

Literature Review

Sr. No.	Title/Author	Overview	Features Considered	Accuracy
1	Machine Recognition of Auslan Signs Using Power Gloves: Towards Large-Lexicon Recognition of Sign Language (Mohammed Waleed Kadous Computer Science and Engineering, University of New South Wales)[2]	The use of Power Glove for detecting hand and wrist movements is proposed in this paper. Instance-Based Learning(IBL) and decision-tree learning are used to process the observed movements. This device is designed to understand approximately 60 Australian Sign Language movements.	Distance, Energy and time, Bounding Box, coordinates, Wrist rotation and finger bend	IBL: 80.6% Decision Tree: 55%

2	COHST and Wavelet Features Based Static ASL Numbers Recognition by Asha Thalange, Dr.S.K.Dixit[3]	The paper proposes two new feature extraction strategies for identification of static signs of numbers 0 to 9 in American Sign Language: Combined Orientation Histogram and Statistical(COHST) Features and Wavelet Features(WF). For the various extraction techniques used, the aim is to achieve full accuracy and then use the best technique for future use. The project expands only the number signs and does not cover the letters and words of ASL.	Orientation Histogram(OH), Statistical Measures(ST), COHST Features and Wavelet Features	OH: 82.92% ST: 74.69% COHST: 87.94% WF: 98.17%

3	Nearest neighbour classification of Indian sign language gestures using Microsoft Kinect camera[4]	The system was built with Microsoft Kinect, which means that ambient light and object colour have a marginal impact on the system's performance. The device proposes a method for recognising Indian Sign Language using the Microsoft Kinect camera in a novel, low-cost and simple-to-use application.	Matlab R2011b, MeshLab, libraries of OpenNI and OpenKinect, Point Cloud Library(PCL)	90.68%

4	Sign Language Recognition Using Convolutional Neural Networks (Sander Dieleman, Pieter-Jan Kindermans and Benjamin Schrauwen ELIS, Ghent University, Ghent, Belgium)[5]	The Microsoft Kinect, Convolutional Neural Networks(CNNs) and GPU acceleration are used in this paper to propose a recognition scheme. The predictive model will generalise to users and environments that were not present during training. This machine is capable of correctly recognising 20 sign language movements.	Depth Map and skeleton Map (created using Kinect), Upper body coordinates and binary image, Higher hand coordinates, CNN	91.70%

5	American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach (Teak-Wei Chong and Boon-Giin Lee, Department of Electronics Engineering, Keimyung University, Daegu 42601, Korea)[6]	Using the Leap Motion Controller(LMC), this study produced a sign language recognition prototype. The LMC is a low-cost, palm-sized portable peripheral device with built-in camera and infrared sensors that is specifically designed to monitor hand and finger motion with high precision in the 3D Cartesian coordinate system. The data is fed to Support Vector Machine(SVM) and Deep Neural Network(DNN) models which is then used to identify 26 letters of ASL.	Hand Palm position, Fingertip position, Distance between fingertips and palm, Angle between arms, distance between fingertips	SVM: 72.79% DNN: 88.79%

6	Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video[7]	The author of the paper describes two extensible systems that use a single colour camera to monitor unadorned hands in real time and use Hidden Markov Models(HMMs) to translate American Sign Language. Instead of attempting a fine definition of hand form, the monitoring stage of the device focuses on the evolution of the gesture over time. Speech recognition and more recently, handwriting recognition have also benefited from HMMs. As a result, they seem to be suitable for recognising complex, structured hand movements such as those used in sign languages.	Hidden Markov Models	Desk mounted camera: 92% User-worn Camera: 98%

7	Recognition of Indian Sign Language in Live Video(Joyeeta Singha Department of Electronics and Communication DBCET, Assam Don Bosco University Guwahati, Assam)[8]	In this paper, a novel method for understanding various alphabets of Indian Sign Language is suggested, which takes into account continuous video sequences of the signs. Preprocessing, Feature Extraction, and Classification are the three phases of the proposed method. Skin filtering and histogram matching are part of the preprocessing stage. The feature extraction stage used Eigenvalues and Eigenvectors and the sign recognition stage used Eigenvalue weighted Euclidean distance. In the video sequences, we found 24 separate alphabets.	Skin Filtering, Histogram Matching, Eigenvalues, Eigenvectors, Eigenvalue weighted Euclidean Distance	96.25%

Table 2.1: Literature Review Table

Chapter 3

Research Gaps and Problem Statement

3.1 Research Gaps

A limited volume of work has been done in this field to recognize the different phenomena that are involved in making sign language recognition more efficient. Also, there is no noticeable work that has been done predominantly for real time gesture recognition by considering various research findings with respect to static and dynamic environments. Most of the techniques used for SLR involve use of hardware components which most of the time affect the cost and thus are not cost-efficient. No such recognizable research has been done to use systems fully based on software. Extensive use of hardware components sometimes inhibits the main objective of using SLR that is to provide a cost-effective option to the deaf and mute community.

3.2 Aim

- To create and train a Sign language Detection Model which can recognize sign gestures and display the predicted text.
- To recognize random sign gestures in real time with high accuracy.

Chapter 4

Proposed System Architecture

Our proposed solution consists of four main steps:

- Dataset Creation
- Image Pre-processing
- Training CNN Model
- Creating a GUI

To achieve a 2-way communication between deaf people and normal people, we have added following features to the system:

- Sign to Speech conversion
- Speech to Sign Conversion
- Word Suggestion and Correction

4.1 System Design and Workflow

A dataset is created by capturing ample amount of images through webcam. When the system starts up, it begins collecting gesture images from the previously generated dataset. The images are then pre-processed and cropped so that only the hand gesture is visible. This is accomplished by drawing contours and determining the contour with the greatest area. The contour region is then photographed and resized to 128 x 128 pixels. These images are then divided into three categories: Training, Validation, and Testing.

After this, the model starts building by adding different layers to the image for the accuracy purpose to all the input images. Then, different layers are added to Model. After the addition of the layers, the training of CNN model begins. With the help of CNN, the model builds a network and creates a prediction model which will be used in the prediction of the result. After this, 'validation' and 'testing' of model is performed to check whether model is over-fitting or under-fitting. Further steps are taken to increase the accuracy of model later on.

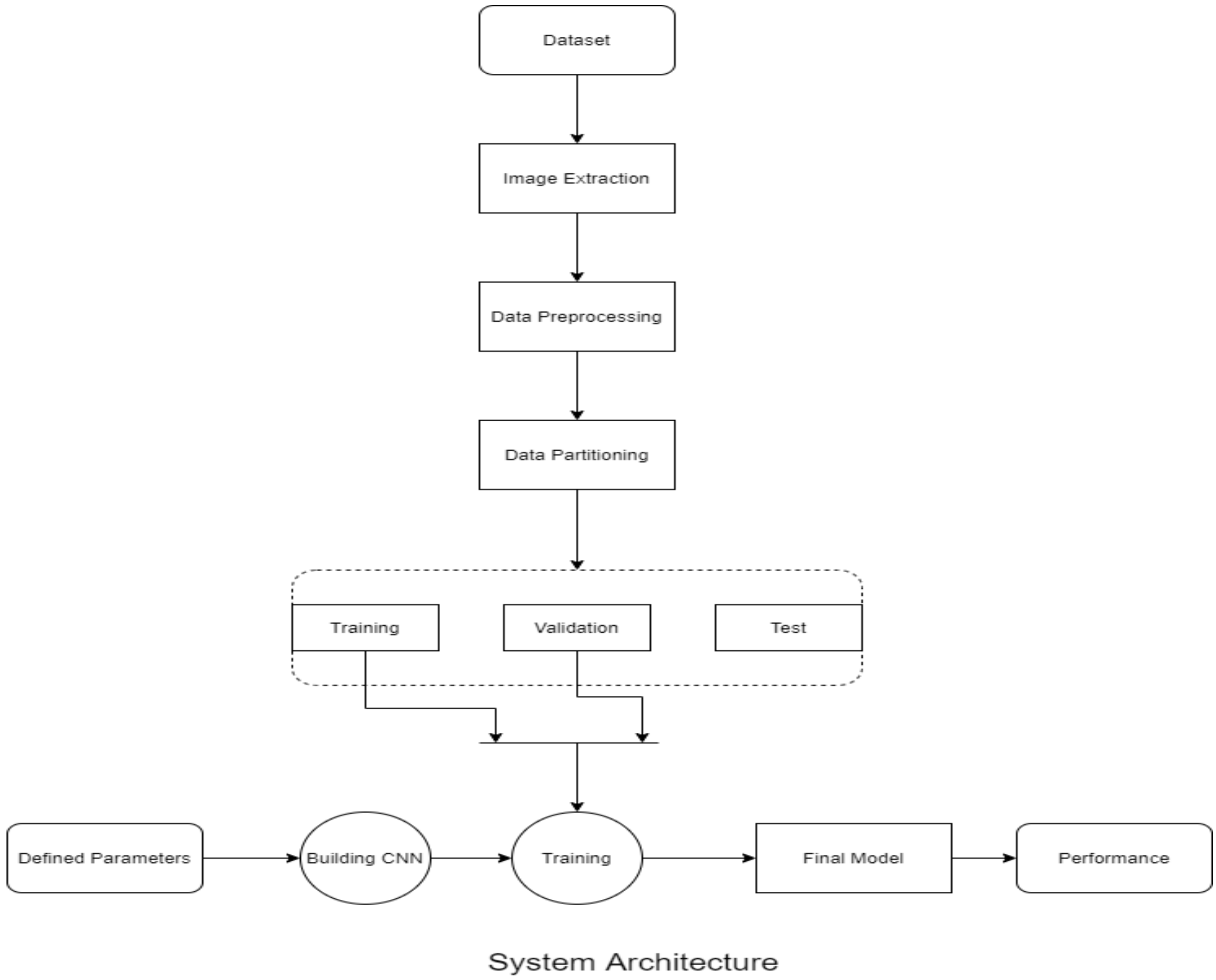

Figure 4.1: System Design

Chapter 5

Implementation

5.1 Dataset creation

We are using OpenCV library to capture images through web-cam. The main objective of using Opencv library is to apply various filters provided by the library. Filters like grayscale conversion and gaussian blur were applied on the target images. Grayscale conversion simply converts the image into black and white image. The input image is in color which is converted into a grayscale image. Gaussian blur removes some of the noise before further processing the image. We are using the 'adaptive-threshold' function to highlight the image borders. The above functions are discussed in detail in further subsections.

We are using Region of interest(ROI) of 300 x 300 pixels for capturing the hand signs. Below is a code snippet of how the gestures in Region of interest are captured.

We have resized all the images to 128 x 128 resolution to minimize the training time of the model. We have created two folders in our dataset, namely Train and Test, both consisting individual folders having 600 images of each symbol.

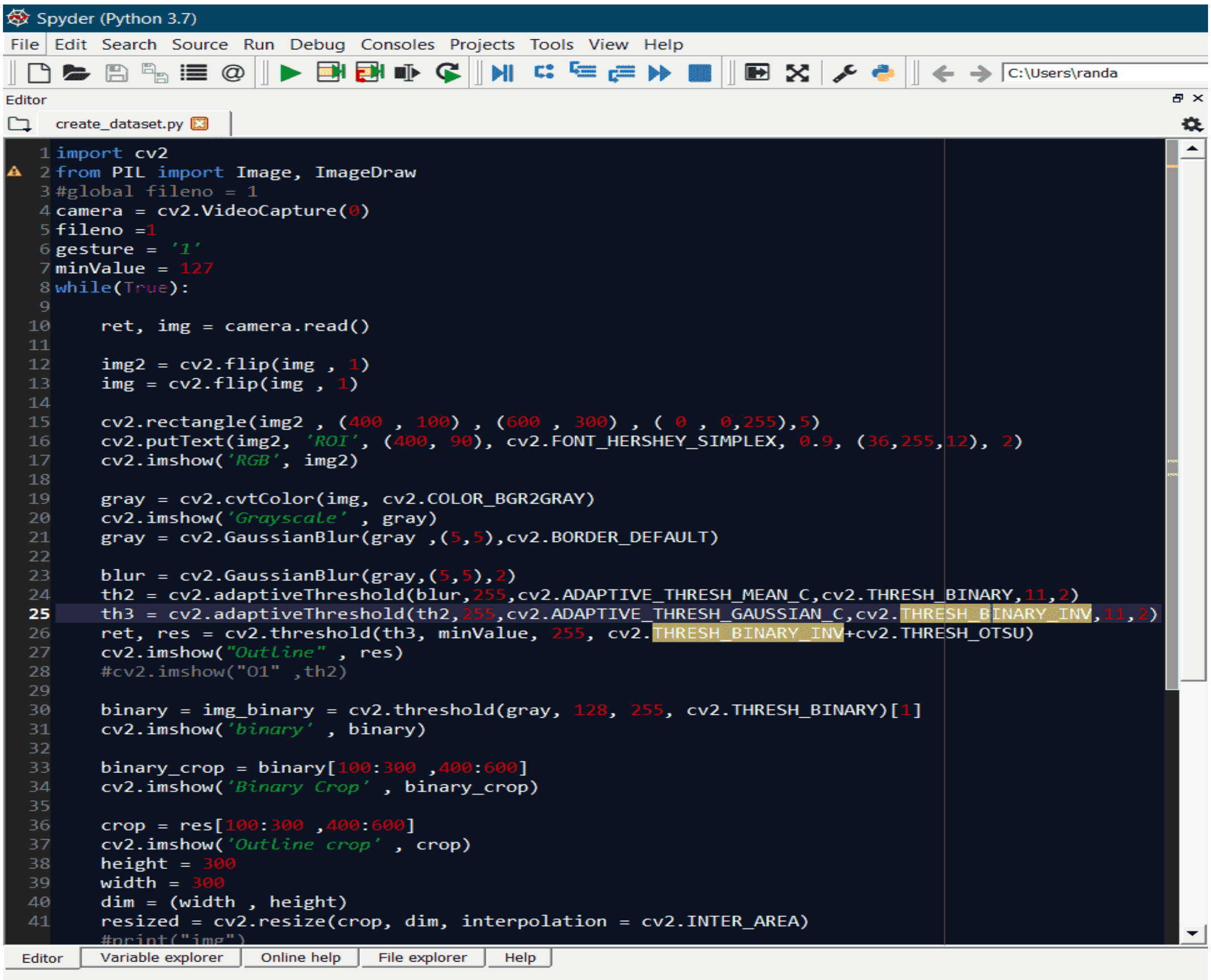

Figure 5.1: A code snippet of Image Preprocessing

5.2 Image pre-processing

We are using two types of images to train the dataset namely binary images and canny edge images. Binary image consists of pixels of only two colours, usually black and white. Canny edge image shows all the edges present in the image which increase the amount of data to be processed but in turn helps in getting useful structural information from an image.

For less symbol classes (10 symbols), using binary images to train the dataset yields good results. But for all 26 English alphabets, canny edge data was found to be more useful than binary data images.

For creating a dataset, we implemented a program and captured the live feed through web-cam.

These steps are followed while preprocessing dataset:

1. Capture a RGB image

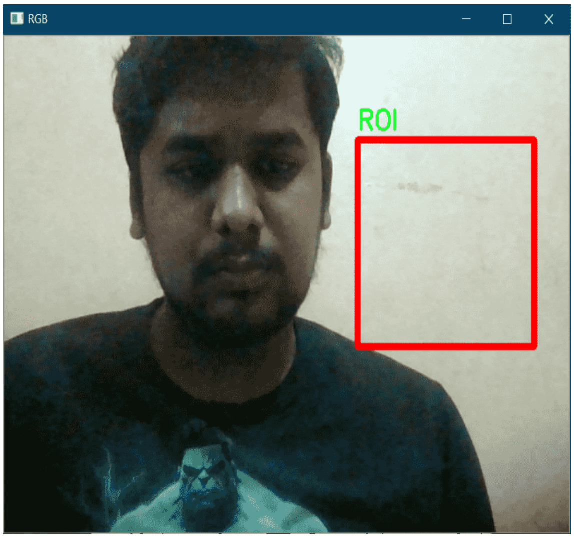

Figure 5.2: RGB Image

2. Convert the RGB image into grayscale and apply gausian blur to it

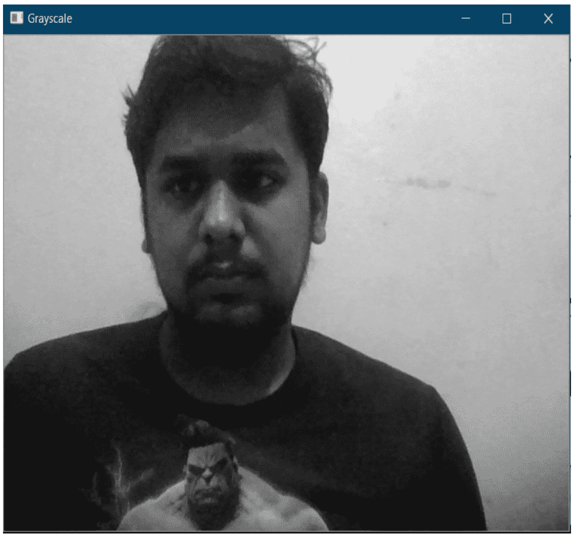

Figure 5.3: Grayscale Image

3. Applying desired threshold to convert grayscale into binary image. Applying a threshold means selecting some specific value for pixels in image and converting the pixels in image such that:

   if (pixel) is less than (threshold) : change the color of pixel to white

   else : change color of pixel to black

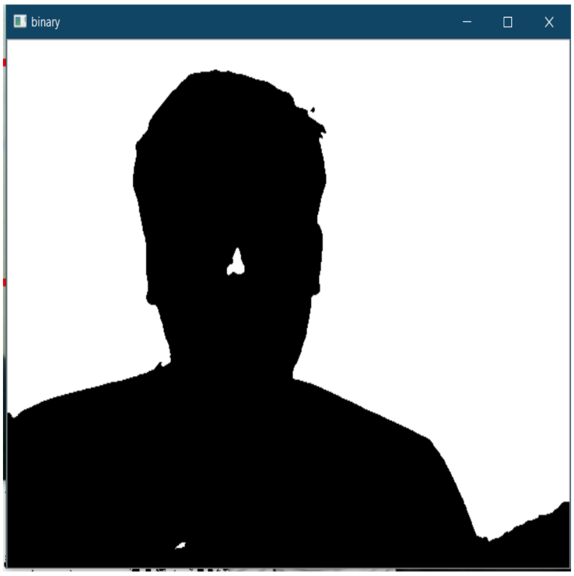

Figure 5.4: Binary Image

4. Cropping and resizing the (Region of Interest)ROI to desired size. In our case, the size of image is 128 x 128 pixels

Figure 5.5: Final Resized Binary Image

5. Converting the grayscale image into canny edge using adaptive-threshold filter

   Adaptive-threshold method calculates threshold value for smaller regions and helps in finding more features in image. This leads to different threshold values for different regions with respect to the change in lighting.

Figure 5.6: Canny Edge Image

6. Cropping and resizing the ROI to desired size. In our case, the size of image is 128 x 128 pixels

Figure 5.7: Final resized Canny Edge Image

7. Saving both images into Local disk for training and testing purpose.

5.3 CNN Model

We are using Convolutional Neural Network(CNN) to train the model. CNN is a deep learning algorithm which takes input as images, assigns weights or biases to various aspects of image which helps in distinguishing the images from one another.

5.3.1 Why CNN?

1. CNNs are automatic feature extractors.

2. We can use max pooling to reduce size of image without compromising with image quality.

3. CNN provide usage of convolutions, which act as feature emphasizer.

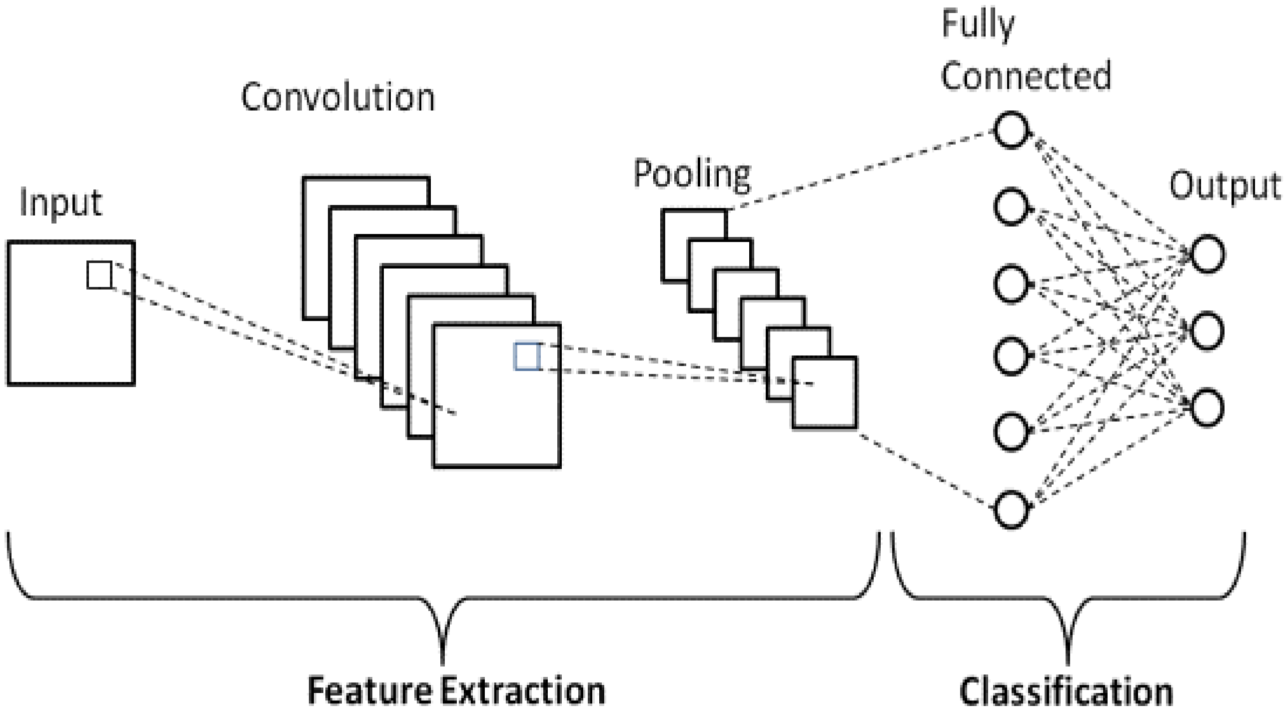

Figure 5.8: Architecture of CNN Model

The OpenGenus IQ[9] team conducted a performance study of various image classifiers in order to determine which was the best classifier for image classification. The findings of an image classification task used to distinguish lymphoblastic leukaemia cells from non-lymphoblastic ones have been presented to support their performance review. The features were extracted using a convolutional neural network and they were fed to different classifiers.

CLASSIFIER	ACCURACY	PRECISION	RECALL	ROC
SVM	85.68%	0.86	0.87	0.86
Decision Trees	84.61%	0.85	0.84	0.82
KNN	86.32%	0.86	0.86	0.88
ANN(for 100 epochs)	83.10%	0.88	0.87	0.88
CNN(for 300 epochs)	91.11%	0.93	0.89	0.97

Figure 5.9: Performance analysis for binary image classification by OpenGenus IQ

5.3.2 Training a CNN Model

In our model, we have added one convolution layer with 32 filters of size (3, 3) which will help in highlighting different features like gesture boundaries in Binary images and minute finger features like emerging thumb between ring finger and middle finger. The output of convolution layer is a feature map which highlights the features of gesture.

We have added a max pool layer, which helps in reducing the size of input data to learning layers of model while ensuring that the details in image are not lost. This is done in order to reduce the computing time.

Lastly, we have 3 fully connected layers with 128, 96 and 64 neurons each and an output layer. This layer learns from the input data and produces weights which are then used to classify signs.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 127008)            0
_________________________________________________________________
dense (Dense)                (None, 128)               16257152
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 96)                12384
_________________________________________________________________
dropout_1 (Dropout)          (None, 96)                0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                6208
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 16,276,389
Trainable params: 16,276,389
Non-trainable params: 0
```

Figure 5.10: Model Summary

5.4 Sign to Speech Conversion

We have implemented Text-to-Speech Conversion using Pyttsx3, a speech engine which converts text to speech. It is python library which works offline and converts Text into Speech. The characters which are recognised using CNN model are grouped together to form meaningful words which are then pronounced using Pyttsx3.

5.5 Speech to Sign Conversion

We are using speech recognition to convert voice commands into text. The text is then fed into a converter function and the results are then displayed through a sequence of images displayed on screen. This is an independent feature and it is provided through a separate GUI.

5.6 Word Suggestion Functionality

Even though we are creating words from recognised characters, it may take some time to spell big words. Also, there is a possibility of incorrect spellings. This is where "Word suggestion and correction" feature comes into play.

We are using US English dictionary to suggest words based on some characters in word. This is done using Hunspell module in python.

5.7 Graphical User Interface

We want a product that allows users to predict sign language in real time. Since the aim is to make it easily accessible without much dependencies, we have created both desktop application and web application.

5.7.1 Desktop Application

This application is created using Tkinter module in Python and has all features of system built in it.

The features include:

- Sign to Text Conversion
- Sign to Speech Conversion
- Speech to Sign Conversion
- Text to Sign Conversion
- Word Suggestions
- Shortcut Mode
- Emergency Mode

The images below display "Sign to Speech/Text Conversion" page and "Text/Speech to Sign Conversion" page.

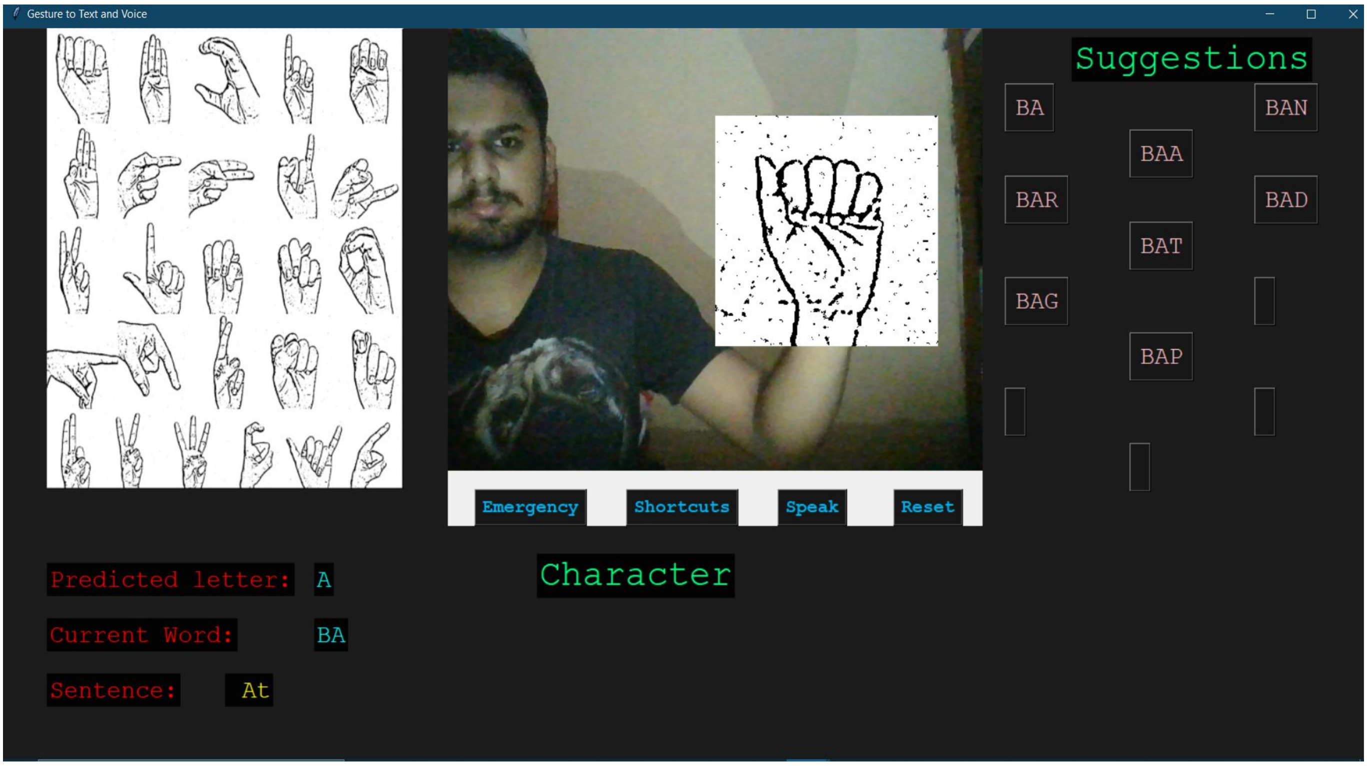

Figure 5.11: Screen 1

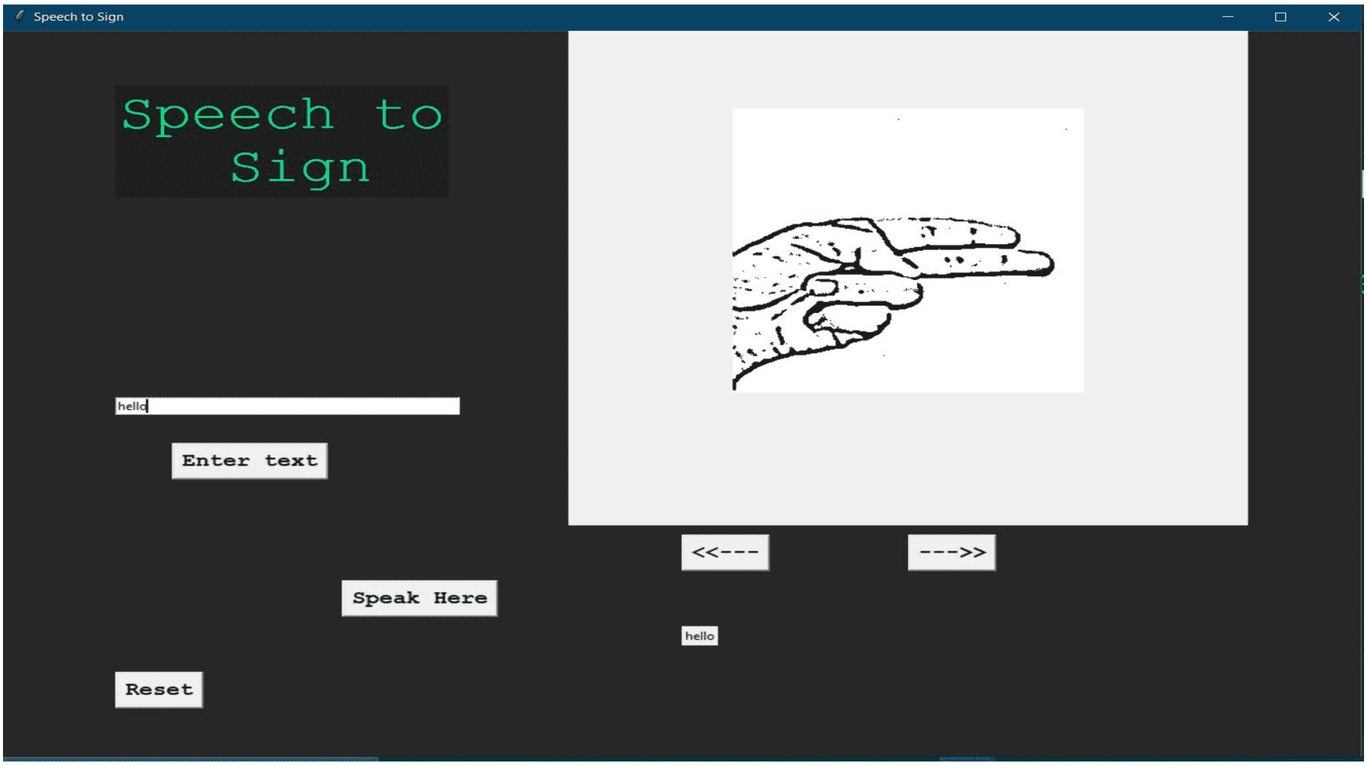

Figure 5.12: Screen 2

5.7.2 Web Application

We have used Streamlit to create the Web application. Streamlit is an open-source app framework for creating and deploying data science applications.

The features included in current version of web-app are:

- Sign to Text Conversion
- Text to Sign Conversion
- Real time Audio-Video Feed

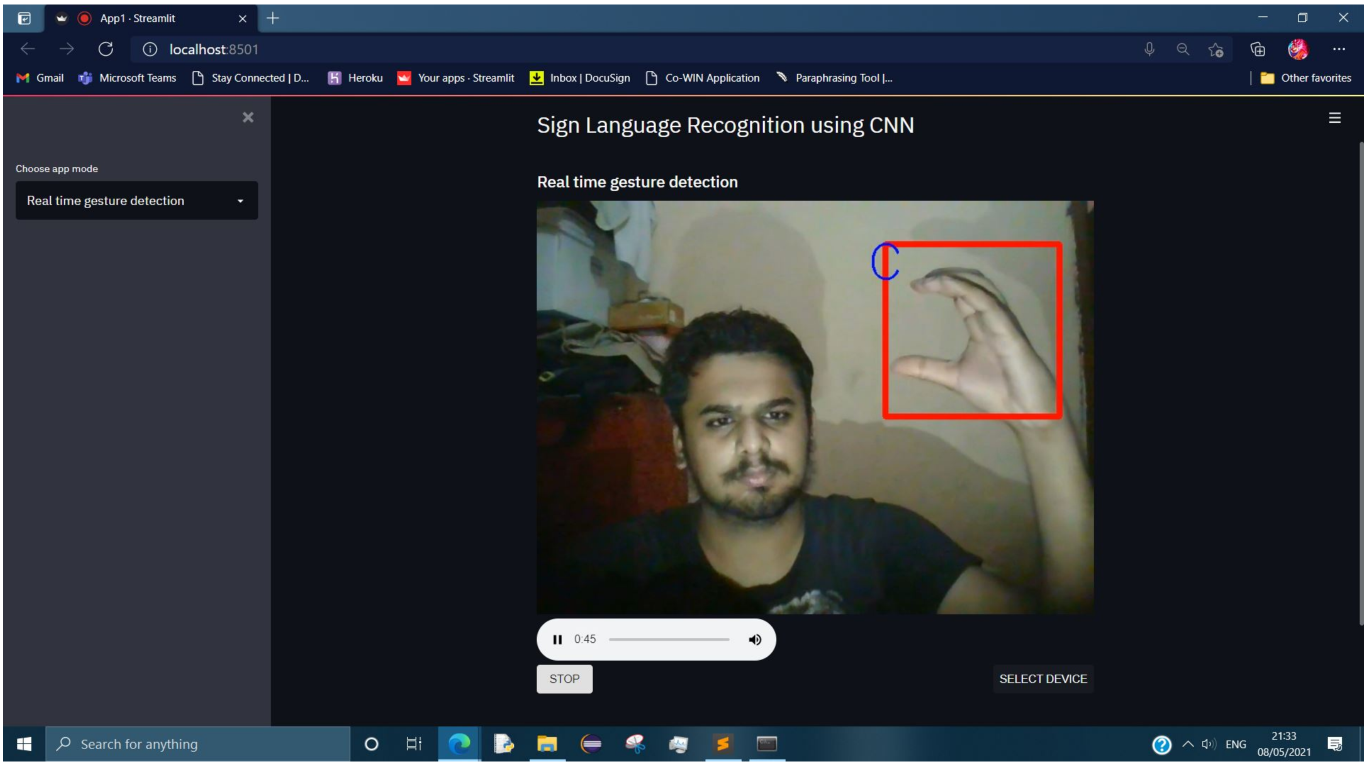

Figure 5.13: Screen 1 of Web App

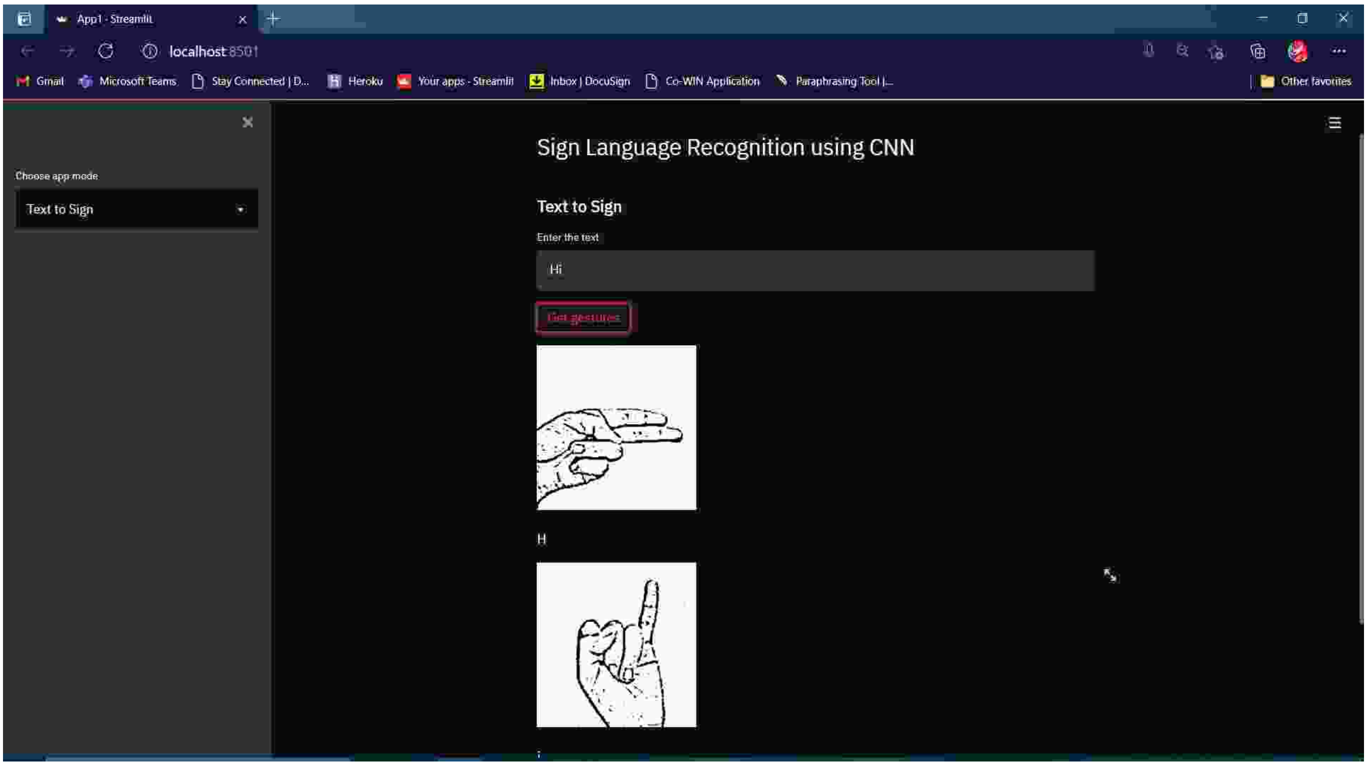

Figure 5.14: Screen 2 of Web App

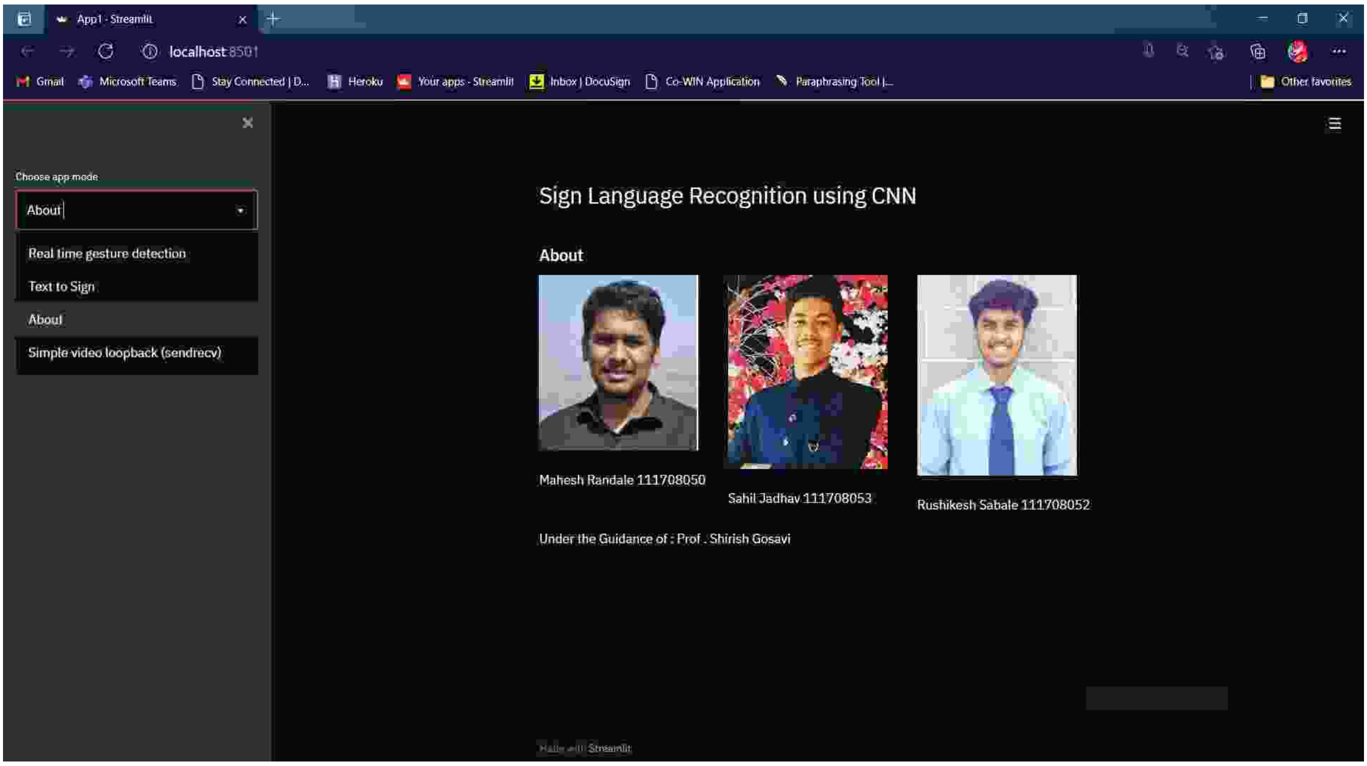

Figure 5.15: Screen 3 of Web App

Chapter 6

Experimental Setup

6.1 Hardware Requirements

1. Web Camera
2. 8 GB Ram

6.2 Software Requirements

1. Tensorflow - A math library based on differentiable programming.
2. OpenCV - Capturing and processing images
3. Keras - A deep learning API written in Python, running on top of the machine learning platform TensorFlow
4. NumPy - For array manipulation
5. Matplotlib - To plot images on graphs
6. Tkinter - To create a desktop application
7. Hunspell - For word suggestions
8. Pyttsx - Text-to-Speech conversion
9. Streamlit - To create a real-time web application

Chapter 7

Results and Discussion

In current CNN Model, we are using one convolution layer and one max pooling layer. We are using two datasets which are:

- Binary Dataset
- Canny Edge Dataset

We trained the model until it had an accuracy of 99% on training dataset. After the training, during validation of version model, we have achieved:

- 98% accuracy while using canny edge dataset
- 100% accuracy while using binary dataset

7.1 Improvements in Binary Dataset

While using binary dataset, the model was over-fitting. We stopped the training phase early after a certain number of iterations to prevent over-fitting.

The model was either providing low accuracy of 80% or was over-fitting, even after adding different methods to avoid over-fitting.

Furthermore, the model was unable to differentiate between signs with slight variations, such as (S, M, N) and so on. Binary images were not capable of showing minute differences in different characters.

Hence we decided to work on canny edge dataset, as it shows a lot of promise for improving the accuracy rate.

7.2 Improvements in Canny Edge Dataset

The canny edge dataset had good accuracy on trained data, but 89% accuracy on test data implied that the model needs to be trained with a variety of input images.

We improved the accuracy of the canny edge dataset by using image augmentation, which allows us to access an infinite number of images. By using max pooling, which shows the most common features of existing feature maps, we were able to increase accuracy even further.

Following these measures, we were able to achieve a 98% accuracy on the canny edge dataset.

Chapter 8

Future Scope

1. Introduce Hand Tracking Algorithm in existing system
2. Extend the model to recognize motion gestures
3. Complete the web application
4. Deploy the Web Application with web hosting platforms like Heroku,Streamlit,et

Chapter 9

Conclusion

In this project, we built a system which will correctly recognize ASL Alphabets and Numbers, which mainly depend on hand and fingers. The model used in this project uses CNN for detection of 26 English ASL alphabets using different image enhancement techniques. This model also uses Convolution layer to highlight the dominant features and max pooling layers to reduce computing cost. The current version of model has one convolution, one max pooling and three fully connected layers. These layers may change while increasing the accuracy of model. The current version of model has accuracy of 98%. Furthermore, added features provide the means for an effortless two way communication between the corresponding entities.

Appendix A

References

1. Using Machine Learning to Interpret Hand Sign Language by Vivek Khurana https://www.youtube.com/watch?v=hfUsL6bUPss

2. Machine Recognition of Auslan Signs Using PowerGloves: Towards Large-Lexicon Recognition of Sign Language(Mohammed Waleed Kadous Computer Science & Engineering, University of New South Wales)

   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.2721&rep=rep1&type=pdf

3. COHST and Wavelet Features Based Static ASL Numbers Recognition - Asha Thalange, Dr. S.K. Dixit https://www.sciencedirect.com/science/article/pii/S1877050916316234

4. Nearest neighbour classification of Indian sign language gestures using kinect camera - ZAFAR AHMED ANSARI and GAURAV HARIT https://www.ias.ac.in/public/Volumes/sadh/041/02/0161-0182.pdf

5. Sign Language Recognition Using Convolutional Neural Networks https://link.springer.com/chapter/10.1007/978-3-319-16178-5_40

6. American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach (Teak-Wei Chong and Boon-Giin Lee

, Department of Electronics Engineering, Keimyung University, Daegu 42601, Korea) https://www.mdpi.com/1424-8220/18/10/3554

7. Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video Thad Starner, Student Member, IEEE, Joshua Weaver, and Alex Pentland, Member, IEEE Computer Society http://luthuli.cs.uiuc.edu/~daf/courses/Signals\%20AI/Papers/HMMs/00735811.pdf

8. Recognition of Indian Sign Language in Live Video(Joyeeta Singha Department of Electronics and Communication DBCET, Assam Don Bosco University Guwahati, Assam) https://arxiv.org/ftp/arxiv/papers/1306/1306.1301.pdf

9. A Comparison study of image classification techniques https://iq.opengenus.org/basics-of-machine-learning-image-classification-techn

10. https://www.tensorflow.org/tutorials/images/cnn

11. https://www.guru99.com/convnet-tensorflow-image-classification.html

12. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

13. Dataset Researches : https://facundoq.github.io/datasets/lsa64/ add