

# Sign Language Detection using Image Processing and Deep Learning

Teena Varma<sup>1</sup>, Ricketa Baptista<sup>2</sup>, Daksha Chithirai Pandi<sup>3</sup>, Ryland Coutinho<sup>4</sup>

<sup>1</sup>Professor, Dept. of Computer Engineering, Xavier Institute of Engineering, Maharashtra, India.

<sup>2</sup>Student, Dept. of Engineering, Xavier Institute of Engineering, Maharashtra, India.

\*\*\*

**Abstract** - Sign Language is a language in which we make use of hand movements and gestures to communicate with people who are mainly deaf and dumb. This paper proposes a system to recognize the hand gestures using a Deep Learning Algorithm, Convolution Neural Network (CNN) to process the image and predict the gestures. This paper shows the sign language recognition of 26 alphabets and 0-9 digits hand gestures of American Sign Language. The proposed system contains modules such as pre-processing and feature extraction, training and testing of model and sign to text conversion. Different CNN architecture and pre-processing techniques such as greyscale, thresholding, skin masking, and Canny Edge Detection were designed and tested with our dataset to obtain better accuracy in recognition)

**Index Terms** - Deep Learning · Convolution Neural Network (CNN) · American Sign Language (ASL) · Canny Edge Detection

## 1. INTRODUCTION

People whose hearing aids are completely or partially damaged are termed as deaf. To communicate with people with good hearing aid the deaf use lip-reading or sign language is a language that employs signs made with the hands and other movements, including facial expressions and postures of the body, used primarily by people who are deaf. There are many different sign languages as, for example, British and American Sign Hand Movements are also sometimes invented by people when they want to communicate with someone who does not speak the same language. The people who have good hearing do not know what is sign language and often lack to communicate with people who are deaf. Communication is a method of exchanging ideas or messages through gesture or text. The hand gesture is a powerful method of communicating our thoughts to others. Gestures can be static or dynamic. Static gestures are involving hand shape and dynamic includes hand movements when communicating.

Sign Language is the most basic and natural way of hearing-impaired people to communicate. Society neglects these physically impaired people and isolates them. To bridge the gap between the normal people and hearing-impaired individuals individual one must know sign language. We aim to develop an efficient system to help get over this barrier of communication. A functioning sign language interpreter can provide an opportunity for a muted to communicate to a non-signing person without the help of an interpreter.

The focus of this work will be to recognize American Sign Language using an android app. The objective is to incentivize algorithms that are ideal enough to be executed on a mobile platform. The system was developed using 26 hand gestures of American Sign Language. The dataset contained approximately 250-300 images per alphabet and 200 images per number. The elementary task was to pre-process the images obtained for this project.

In this paper, the raw images procured were pre-processed using the following algorithms- 1. Grey scaling 2. Skin Masking 3. Threshold 4. Canny Edge Detection.

Post Pre-processing the images were fed into the Convolution Neural Network model for training and testing purposes. Various CNN architectures were performed and tested on our dataset to identify the best architecture for the recognition of the hand gestures.

## 2. EXISTING SYSTEM

Priyanka C Pankajakshan and Thilagavati B [6] have proposed a system for sign language recognition using ANN (Artificial Neural Network). The system consists of live capture of hand gestures to process and identify the sign using ANN. Das, A., Gawde, S., Suratwala, K., & Kalbande .D. [1] and Rao, G. A., Syamala, K., Kishore, P. V. V., & Sastry. [2] has performed fundamental research on sign language datasets using the CNN algorithm to achieve satisfactory results from the training and testing of the dataset. In [1] the authors proposed a system using CNN's Inception v3 on the dataset to test its accuracy as well and found it to be better than CNN. In [2] the proposed system is using selfie language to process the images and is tested using stochastic pooling. The CNN model used to train the dataset in [2] was performed using different window sizes and different batches of images and output accuracy achieved by them were 92.88% compared to the other methods they researched.

Mahesh Kumar NB [3] is using MATLAB to perform feature extraction on the dataset used and then performing Linear Discrimination Analysis (LDA) for gesture recognition. Bantupalli, K and Xie, Y [4] use the CNN model named Inception to extract spatial features from the video stream for Sign Language Recognition (SLR). Then, by using a LSTM (Long Short-Term Memory), a RNN (Recurrent Neural Network) model, they [4] extract temporal features from the

video sequences via two methods: Using the outputs from the Softmax and the Pool layer of the CNN respectively.

Anup Kumar, Karun Thankachan and Mevin M. Dominic [5] have developed a system using Support Vector Machine. The images used in [5] are pre-processed using skin segmentation and extract appropriate features from the image obtained. This pre-processing of images is completed by converting the image to greyscale and then performing HSV thresholding [5].

### 3. PROPOSED SYSTEM

The below module is fed an image as an input that delivers the alphabet/number corresponding to the sign visible in the image/video.

Four major steps are carried out by the process as follows:

1. Pre-processing the image
2. Feeding the image into the CNN model
3. Displaying the predicted text for the image passed.

In the first step, the input is obtained as an image or captured from a video. The captured image is then pre-processed using the various methods mentioned in section IV and forwarded to the CNN model. The CNN model tests the loaded image against the trained images and predicts the sign with the most probable labels from the already trained model.

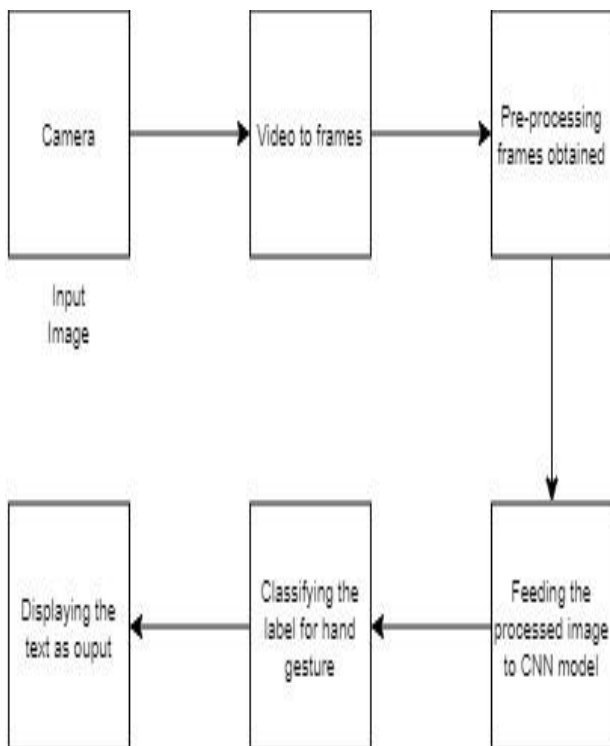


Fig-3.1: High level system architecture

### 4. IMPLEMENTATION

CNN model cannot accurately predict labels when unprocessed images are fed directly to it. The main issue with the processing using CNN is its inability to cancel out the background properly. Hence, the images are required to be processed separately using various image processing techniques. The images were resized to 45x45 size before processing. The numerous techniques applied to the images in our dataset are described below.

#### 4.1 Grey Scaling

The first method we used to process our images was by just converting the RGB image to Greyscale and resizing the image to pass through the CNN model used to train the images.

Grayscale is a range of consistent monochrome shades from black to white. Digital images can be saved as or black and white images, even color images comprise of grayscale information. Each pixel contains a luminance value, regardless of its color. Luminance for images can be characterized as brightness or intensity, which can be measured on a scale from black to white.

$$\text{Pixel} = 0.299 * \text{Red} + 0.587 * \text{Green} + 0.114 * \text{Blue} \quad (1)$$



Fig- 4.1: Output of Grayscale

#### 4.2 Skin Masking

The first step of skin masking is to convert to greyscale as done in section 4.1. The next step is converting RGB to HSV. The images are converted to grayscale and then to HSV and are then given to the skin masking inbuilt function.

The color space of an image can be defined using HSV (Hue, Saturation, and Value). In HSV, the hue denotes the color. In this model, the hue is an angle from 0 degrees to 360 degrees. Saturation determines the range of grey in the color space. The brightness of the color that varies with color saturation is identified with value.

This skin masking removes the extra background and helps to focus on the hand region needed to be used. According to the output of HSV we set the upper and lower bound for the image. Below is the flowchart of steps carried out to perform skin masking.

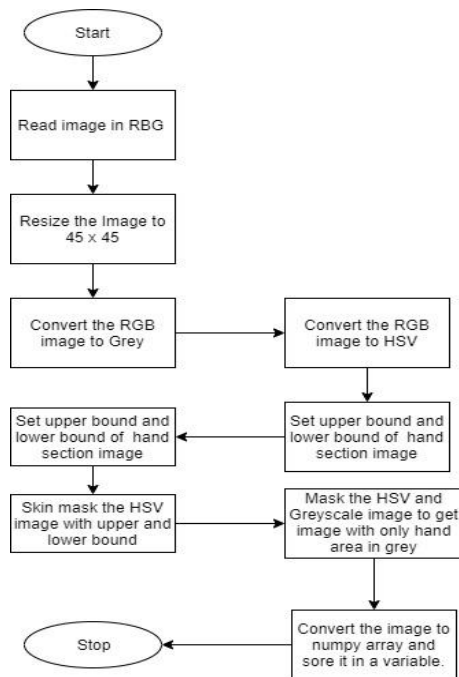
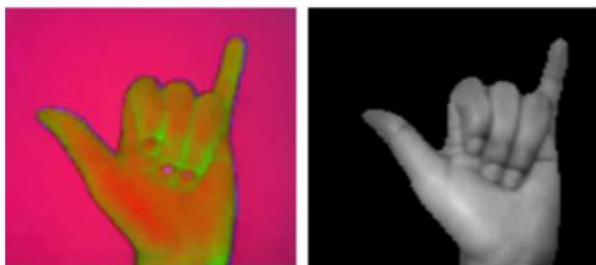


Fig-4.2: Flowchart of skin masking

In our case, we set the lower bound to [0,40,30] and the upper bound to [43,255,254]. These bounds are set for only the hand area of the image and segment the HSV image with the help of the bounds. Subsequently, we mask the greyscale and the segmented HSV image to get the final image.



INPUT OUTPUT

Fig-4.3: Output of Skin masking

#### 4.3 Threshold

Image thresholding is a simple way of dividing an image into the forefront and background. This image analysis technique is a type of image segmentation that divides objects by converting grayscale images into binary images. Image thresholding is effective to process an image with greater levels of contrast. There are different types of threshold, for our study we have used Threshold Binary Inverse and Threshold Otusu. The threshold value for the processing is set to [127,255]. The below image Fig. 4.4 is obtained after applying the above-mentioned threshold



INPUT OUTPUT

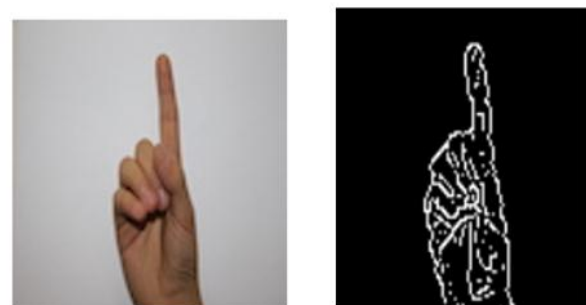
Fig-4.4: Output of Threshold

#### 4.4 Canny Edge Detection

Canny edge detection is a technique to extract useful architectural information from different vision objects and adequately reduce the amount of data to be processed. The criteria for edge detection incorporate:

1. The detection of edges should accurately capture as many edges shown in the image with a low error rate
2. An edge in the image should only be marked once, to avoid image noise and avoiding the creation of false edges.

Canny edge detection was applied to images after following the process of skin masking Fig.4.5 on the images to get defined edges of hand gestures and eliminating the background.



INPUT OUTPUT

Fig-4.5: Output of Canny Edge Detection

#### 4.5 Convolution Neural Network

We have designed our CNN algorithm by the knowledge perceived from [1][2]. CNN's are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of artificial intelligence by offering systems that simulate different types of biological human brain activity. The CNN model is fed with the processed images to classify the images.

We are using the American Sign Language alphabet dataset to perform the image processing techniques mentioned in section III. The method used to train the images is Convolution Neural Network [2]. We feed the CNN model with the processed images to classify the images. Once the model is fitted, we use it to predict random images from the computer to predict the text and display it on the screen.

This CNN architecture is implemented on the processed images. The results obtained by the various models are illustrated below in section 5. The scope of the model is discussed in Section 6.

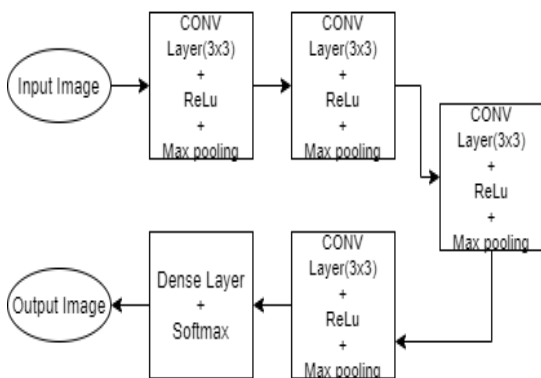


Fig- 4.6: Proposed Deep CNN architecture

Two different CNN model architecture was implemented by us. The first proposed CNN architecture uses 32,64,64,128-bit filter layers respectively with window size 3x3, activation function ReLu (Rectified Linear Unit) and Max pooling for each layer. The classification stage is implemented with dense/fully connected layers followed by activation function SoftMax. The second model architecture works with 16,32,64,64 – bit filters layers for each layer designed and the rest architecture is similar to the above model. The first model was found more accurate in the first attempt; hence the further system was developed using that model architecture.

Various interpretations were used to measure the performance of the models. Before that here are the basic terms needed to understand-

True Positives (TP) - The correctly predicted positive values which means that the value of actual class and the predicted class is true.

True Negatives (TN) - The correctly predicted negative i.e. wrong values which means that the value of actual class and predicted class is false.

False Positives (FP) – When actual class is false and predicted class is true.

False Negatives (FN) – When actual class is true but predicted class in false.

Once we understood and learned these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.

Accuracy - Accuracy is the most accurate performance measure and it is a ratio of correctly predicted observation to the total observations

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (2)$$

Precision - Precision is the ratio of accurately predicted positive to the total predicted positive observations.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

Recall (Sensitivity) - Recall is the ratio of accurately predicted positive observations to the all observations in actual class

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

F1 score - It is calculated using Precision and Recall. Therefore, f1 score takes both false positives and false negatives into account. f1-score works best if false positives and false negatives have similar cost.

$$F1\ score = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (5)$$

The above discussed measures were used on the model discussed in this section to measure the performance of the CNN models implemented.

## 5. EXPERIMENTAL EVALUATION

The result obtained by training the CNN model on different images obtained after applying the image processing techniques mentioned in Section 3 is discussed in this section. This section highlights the output results with the use of graphs for a better understanding of the accuracy obtained during the training and testing of the dataset.

### 5.1 CNN using Grayscale images

The below accuracy graph Fig.5.1 shows the accuracy of every epoch. The accuracy graph was obtained by pre-processing the images using the method mentioned in Section 4.1. The training accuracy given at the 5th epoch is 90% and the testing accuracy is 91%. The model was saved

and used for predicting a random number of images from the dataset.

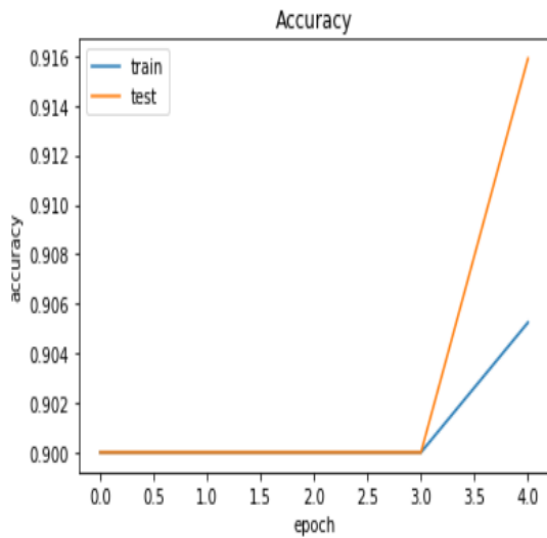


Fig-5.1: Accuracy graph of CNN using grayscale images

### 5.2 CNN using Skin Masking

The below accuracy graph Fig.5.2 shows the accuracy of every epoch. The accuracy graph was obtained by pre-processing the images using the method mentioned in Section 4.2. The training accuracy given at the 5th epoch is 95% and the testing accuracy is 96%. The model was saved and used for predicting a random number of images from the dataset.

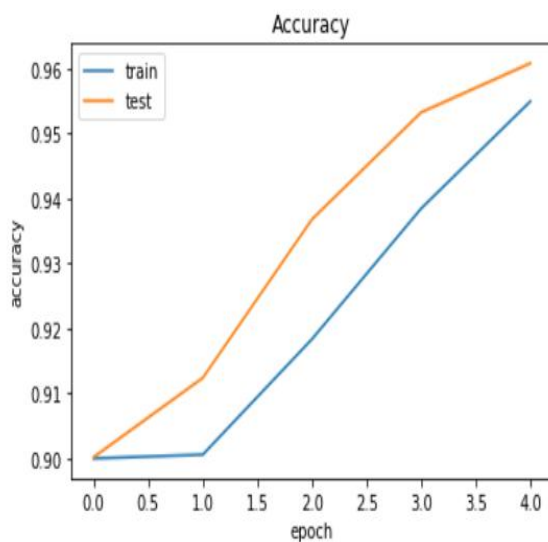


Fig- 5.2: Accuracy graph of CNN using skin masking

### 5.3 CNN using Thresholding

The below accuracy graph Fig.5.3 shows the accuracy of every epoch. This graph was obtained by using the pre-

processing technique discussed in Section 4.3. The training accuracy given at the 5th epoch is 96% and the testing accuracy is 97.25%. The model was saved and used for predicting a random number of images from the dataset.

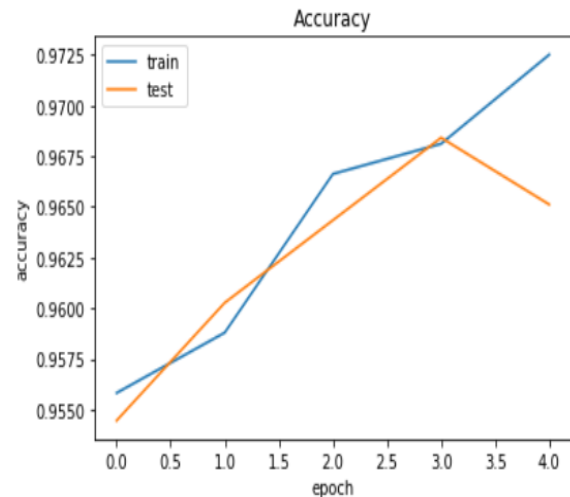


Fig- 5.3: Accuracy graph of CNN using threshold

The Fig.5.4 shows the precision, recall, f1-scores and support measures discussed in Section 4.5 for each label numbers 0-9.

	precision	recall	f1-score	support
0	0.87	0.94	0.91	51
1	0.95	1.00	0.98	42
2	0.96	0.96	0.96	54
3	0.93	1.00	0.97	43
4	0.91	0.83	0.87	63
5	0.98	1.00	0.99	61
6	0.90	0.93	0.91	46
7	0.91	0.91	0.91	46
8	0.96	0.89	0.93	57
9	0.96	0.92	0.94	53
accuracy			0.94	516
macro avg	0.94	0.94	0.94	516
weighted avg	0.94	0.94	0.94	516

Fig- 5.4: Accuracy graph of CNN using threshold

### 5.4 CNN using Canny Edge Detection

The below accuracy graph Fig.5.5 shows the accuracy of every epoch. This graph was obtained by using the pre-processing technique discussed in Section 4.4. The accuracy last given at the 5th epoch is 98% and the testing accuracy is 98.3%. The model was saved and used for predicting a random number of images from the dataset.

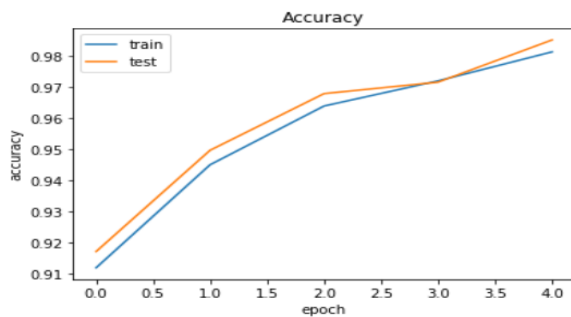


Fig- 5.5: Accuracy graph of CNN using Canny Edge Detection

The Fig.5.6 shows the precision, recall, f1-scores and support measures discussed in Section 4.5 for each label numbers 0-9.

	precision	recall	f1-score	support
0	0.94	0.96	0.95	47
1	0.98	0.85	0.91	52
2	0.87	0.74	0.80	53
3	0.98	0.94	0.96	53
4	0.93	0.69	0.79	54
5	0.98	1.00	0.99	55
6	0.80	0.86	0.83	56
7	0.79	0.88	0.83	51
8	0.77	0.90	0.83	52
9	0.81	1.00	0.90	43
accuracy			0.88	516
macro avg	0.88	0.88	0.88	516
weighted avg	0.89	0.88	0.88	516

Fig- 5.6: Precision, recall f1-score and support of CNN using Canny Edge Detection.









Sr no.	Input test image	Output test image	Actual output	Output obtained
Greyscale			7	3
Skin mask			7	4
Threshold			7	7
Canny edge detection			7	7

Table – 1: Comparison table

The above table shows the comparison carried out for predicting a label for an input image for number 7 using the above-mentioned CNN models.

From the table we can observe that the models using threshold and canny edge detection predict the label of the image passed to it as desired i.e. number 7 whereas the models using greyscale and skin masking do not predict the correct gesture.

As discussed in this section, the accuracy for CNN using threshold and Canny edge detection has higher training and testing accuracy than the others and predicted the hand gesture accurately as observed in Table 1. The result obtained by two methods through live detection are shown below. Live detection was carried using Python OpenCV libraries to predict the output of the image obtained through webcam. The output shows the processed image part as well wherein, you can view how the image is looked after being captured and processed.

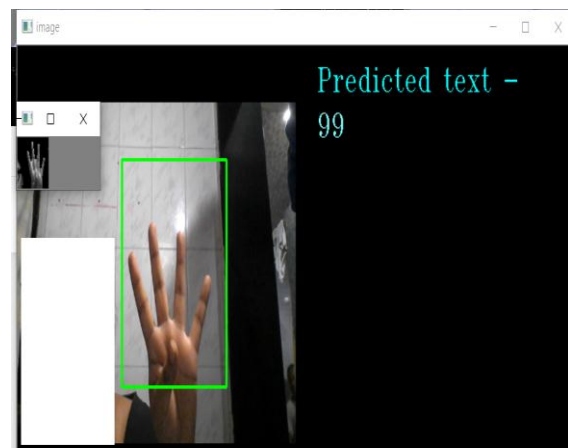


Fig-5.7: Live detection of gesture using Skin masking

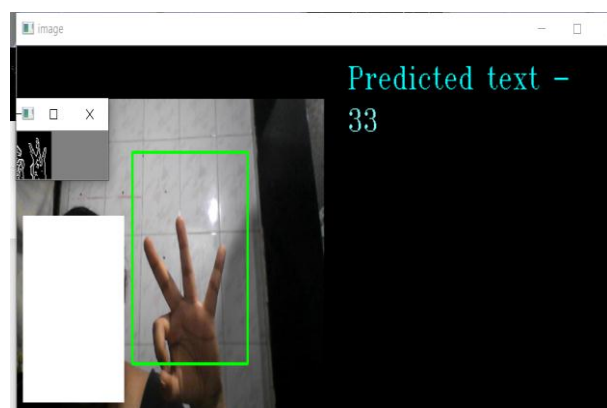


Fig-5.8: Live Detection of gesture using Canny Edge detection.

Hence, we are going to use the CNN model by pre-processing the images by canny edge detection technique and still try to further develop a better model using these results.

## 6. FUTURE SCOPE

The future scope of this project is can be vast as the development of a system for sign language is been evolving research. The model currently is limited to only the American sign language; thus, it is expected to be enhanced enough to work on Indian sign language. The model requires enhancement on fields of capturing dynamic gestures as well because currently the model can only predict or interpret for static finger spellings. The datasets are also in need to be enhanced with better and accurate images. The images can be modified by adding pictures with different light density. Further training of the model to achieve efficient detection for two hand gestures. Sign language doesn't just consist of alphabets and digits, there are words also present in sign language.

Android Application- that can be used by anyone, anywhere to communicate with people who are deaf and dumb and understand sign language. The deployment of the model to an application would make the idea of bridging the gap for communication between normal people and the hearing-impaired people. Hence the development of an efficient and easy operable app is underway. The App would contain a module to convert live captured images to text and vice versa. One can't communicate without understanding the words or just by the means of an alphabet. To overcome this, the model needs continuous training to detect words and common expressions.

## 7. CONCLUSION

Our project objective is to bridge the gap by introducing an economical computer application in the communication path so that the sign language can be automatically captured, recognized and translated to text for the benefit of deaf people. The image obtained must be analyzed, processed and converted to either sign or textual display on the screen for the benefit of the hearing impaired. We have learned and demonstrated that CNN can learn how to identify and predict the text. We have created a model that pre-process the image to the required nature for it to be fed into the model. The system is an approach to ease the difficulty in communicating with those having speech disabilities. The amount of training and validation loss observed with the proposed CNN architecture was less. We have tried different

image processing techniques to find the best one we need for our use. During the live capture testing, the Canny edge algorithm with accuracy 98% demonstrated better results than the other techniques discussed in Section 4 and Section 5.

## REFERENCES

- [1]. Das, A., Gawde, S., Suratwala, K., Kalbande, D. "Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images". 2018 International Conference on Smart City and Emerging Technology (ICSCET), (2018).
- [2]. Rao, G. A., Syamala, K., Kishore, P. V. V., Sastry, A. S. C. S. "Deep convolutional neural networks for sign language recognition", (2018).
- [3]. Mahesh Kumar NB. "Conversion of Sign Language into Text". International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9, (2018).
- [4]. Bantupalli, K, Xie, Y. (2018). "American Sign Language Recognition using Deep Learning and Computer Vision". IEEE International Conference on Big Data (Big Data), (2018).
- [5]. Kumar, A., Thankachan, K., & Dominic, M. M. "Sign language recognition". 3rd International Conference on Recent Advances in Information Technology (RAIT), (2016).
- [6]. Pankajakshan, P. C., & Thilagavathi B.. Sign language recognition system. International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), (2015).