

Sign Language Recognition using Convolutional Neural Network and Long-Short Term Memory

Jidnyesha Gaikwad¹, Riya Jalindre², Samruddhi Kalyankar³, Ruchira Walkikar⁴ and Prof. Mrs. M.R Apsangi⁵

¹⁻⁵PVGCOET and GKPIoM, Pune-411009, Maharashtra, India

Email: jid2301jiggs@gmail.com; riyajalindre24@gmail.com; samruddhikalyankar1920@gmail.com; rwalkikar09@gmail.com; mra_it@pvgcoet.ac.in

Abstract—The system "Sign Language Recognition using Convolutional Neural Network(CNN) and Long-Short Term Memory(LSTM)"aims at building a Deep Learning model using Computer Vision to recognize various signs used in American Sign Language(ASL). This system helps to establish a communication edge and understanding of a common language. People with some hearing disabilities need a special sign language to communicate. The system intends to reduce these difficulties. Our project consists of an automatic recognition system for hand gestures in ASL and furthers its conversion to text and speech. The idea deals with images of bare hands, which allows the user to interact with the system in a natural way. It provides an opportunity for visually im- paired people to communicate with normal people without the need for an interpreter. In this autonomous model classification, Deep Learning algorithms are trained using a collection of image data and tested using validation image dataset. This image dataset of different signs in ASL is downloaded from Kaggle and loaded to be used by the model. For the image dataset, binary images are used, which gives better results than the datasets used in previous models. Two Deep Learning algorithms are applied to the dataset which includes a CNN for classification and a LSTM network to store temporal features of the data used for sentence formation. The model specified in this article is used for recognizing gestures of A-Z alphabets, 0-9 digits, and also forms sentences using the recognized gestures of alphabets.

Index Terms— Sign Language Recognition, cnn, lstm, asl, opencv, python, image processing.

I. INTRODUCTION

Speech and language are the primary media of communication for the human race which enables them to move about with their everyday activities with ease. However, the vocally and hearing-impaired population is deprived of this medium thus making it difficult for them to completely be a part of the dominant unimpaired society. The majority of the population uses spoken language as a medium of communication while a group of specially disabled people who are not able to speak and listen, use sign language to communicate. Sign languages do not have any verbal connection to the maturity of spoken languages. Other people realize it is troublesome to grasp and interpret the meaning of sign language, expressed by hearing/verbally impaired people. Few technologies help to connect this group of specially-abled people to the rest of the world.

Hearing-disabled people use sign language to express their studies and feelings and support information delivered in diurnal exchanges. Sign language recognition aims to describe the significant movements of the mortal body, especially hands, dissect them and understand them. Similar systems may come life-saving when

disabled people are in helpless situations like heart attacks, accidents, etc. Spoken sign languages also develop naturally as a result of groups of people interacting with one another; regions and cultures also play an essential role in their development. Common sign languages are American Sign Language (ASL), British Sign Language (BSL), and French Sign Language (FSL). Although the use of language for communication has helped break problems, it frequently faces challenges as well. For case, effective communication generally requires that all involved parties are able to apprehend and acknowledge at least one common language. This isn't always the case; in specific cases, communicating parties may calculate on different written emblematic, sign, or oral languages.

A system designed to interpret sign gestures would be a great help for people as English is a standard language. The sign language-to-text conversion module is a hand gesture recognition module. Hand-gesture recognition involves recognizing the dynamic gesture by comparing it with pre-defined metrics of the trained models. Hand-gesture recognition systems find their application in areas such as character recognition, home automation, and much more.

II. LITERATURE SURVEY

Qazi Mohammad Areeb et al., [1] designed two models for classification, of which one is an object detection model applied after extracting frames. The first model consists of a three-dimensional convolutional neural network(3D CNN), while the second comprises a pre-trained Visual Geometry Group(VGG-16) and recurrent neural network(RNN) with an LSTM scheme.

Disha Shah et al. [2], used dual-handed Indian foreign language as a collection of images and interpreted the sign using Python. The authors have used CNN to train the model. The system was trained for 2000-3000 images with increasing epochs in order to increase the accuracy.

Dhulipala [3], Two models were developed by the authors of this paper, Convolutional Neural Network and Long-short Term Memory, and their performance was calculated using a multi-class confusion matrix.

The purpose of this survey is to view the trends in Computer Vision and scrutinize different approaches toward the design of a Sign Language Recognition System. As discussed above Q. M. Areeb, et.al [1] designed two models for classification which uses LSTM to store temporal feature. Also in Disha Shah, et.al [2] designed a system that used dual-handed gestures and CNN to recognize them. It is clear from the survey that the use of both CNN and LSTM is most in benefit of the composition of the system.

III. SIGN LANGUAGE RECOGNIZER

To solve the sequential processing of the signs performed we modeled together, the CNN with LSTM. Following figure 3.1 illustrates the detailed architecture of our system.

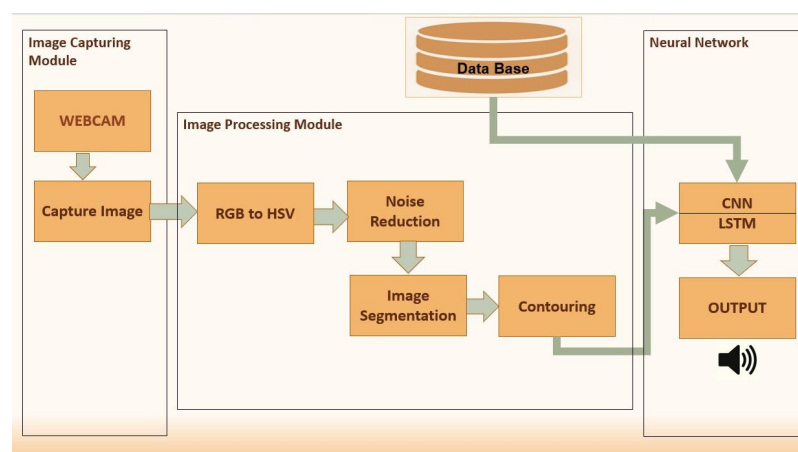


Fig: 3.1 System Architecture

The recognizer incorporates three utmost modules and a database that accommodates the images to be fed as inputs for training purposes. The modules namely are the image capturing module, an image processing module, and a module for the neural network. The modules are further explicated in detail.

A. Image Capturing Module

In Sign Language Recognition, image capturing is an important module in which the sign language gestures are converted into a textual format or machine-readable format. Several factors like lighting conditions, camera angle, and back-ground affect the accuracy of the model. Lighting should be bright enough to ensure clear and visible images. The camera angle should be such that it should be able to capture the hand gestures without obstructing any part of the body of the user performing the signs. The background should be plain and shouldn't have any objects apart from the user, to avoid any distractions from the sign language gestures.

Different types of cameras and sensors can be used to capture images, such as RGB cameras, depth cameras, web cameras, and wearable sensors. Each of these devices has its advantages and disadvantages, depending on the application and the specific requirements of the Sign Language Recognition System. We used a web camera in our project.

A user performs signs in front of the camera and the sign gesture is captured by the web camera in the form of a live feed. Frames are extracted from this live feed and it undergoes processing to give the text and voice corresponding to that gesture. The image processing data will be processed in real time and not be stored.

B. Image Preprocessing

RGB to HSV conversion

The image captured by the webcam is in RGB color model and is reused to convert it into a black and white image. The hand gesture is noted in the image and those values of achromatic and intensity values are taken and reused. In order to convert an image from RGB to HSV color space, the following way can be taken:

- Convert the RGB values to a scale from 0 to 1 by dividing each value by 255.
- Find the outside and minimum values among the R, G, and B factors for each pixel in the image.
- Calculate the difference between the outside and minimum values to determine the range of the color factors. This is called the hue.
- Calculate the tinge(H) element using the formula:
 $H = 60 * (G - B) / \text{hue}$ (if maximum = R)
 $H = 60 * (B - R) / \text{hue}$ 120 (if maximum = G)
 $H = 60 * (R - G) / \text{hue}$ 240 (if maximum = B)
(Note If the hue is 0, the tinge is undetermined.)
- Calculate the achromatic(S) element using the formula $S = \text{hue} / \text{maximum}$
(Note If the value of maximum is 0, the achromatic is 0.)
- Calculate the value(V) element using the formula $V = \text{maximum}$

Noise Reduction

Digital images are susceptible to colorful types of noise. Noise reduction is an important task in image processing, as noise can degrade the quality of an image and make it delicate to prize useful information from it. The model specified in our system uses two noise-filtering techniques to get a smoother image:

Gaussian Blur : Gaussian blur is an image processing technique used to smooth or blur an image by reducing the high-frequency components in the image. In the context of image processing, the Gaussian blur works by convolving the image with a Gaussian kernel. When the kernel is applied to the image, each pixel in the image is replaced with a weighted average of its neighboring pixels, with the weights determined by the Gaussian function. The amount of blur applied to the image is controlled by the size of the kernel. A larger kernel will result in a stronger blur, while a smaller kernel will produce a more subtle effect. The background pixels of the input image is removed in Gaussian Filter. An OpenCV function `cv2.gaussianBlur()` is used to for Gaussian Smoothing of the RGB to HSV converted image. Below is the syntax of `gaussianBlur()` function :

`dst = cv2.gaussianBlur(src, ksize)`

where src is the HSV image and ksize is the kernel size(11x11)

Median Filtering : The technique of median filtering is based on re- placing the pixel value at each location with the median value of its neighboring pixels. After applying Gaussian Blur, the image is further denoised using the Median Filtering algorithm. An OpenCV function `cv2.medianBlur()` is used for the further blurring of the Gaussian Filtered image. Below is the syntax of `median- Blur()` function :

`dst = cv2.medianBlur(blur, ksize)`

where blur is the output image of Gaussian Filter and ksize, is the window size(15) must be odd and greater than 1.

Image Segmentation

Image segmentation is a computer vision technique that divides an image into multiple regions or segments, each representing a meaningful part of the image, such as an object, background, or foreground. Thresholding is a simple and effective method that involves setting a threshold value to separate the foreground and background of an image. Our system uses a combination of two thresholding techniques to get a smoother image. The two algorithms used are:

Otsu Thresholding: The basic idea behind Otsu thresholding is to find an optimal threshold value that minimizes the intra-class variance of the foreground and background pixels. The threshold value is determined by maximizing the between-class variance, which is the variance between the mean values of the two classes. To perform Otsu thresholding:-

1. First compute a histogram of the image intensity values, which provides an estimate of the probability density function of the image.
2. Then normalize the histogram so that the sum of all bin values is equal to one.
3. Next, the algorithm iteratively calculates the between-class variance for each possible threshold value and selects the threshold value that maximizes this variance.
4. Once the threshold value is determined, all pixels in the image with intensity values above the threshold are assigned to the foreground, while those with values below the threshold are assigned to the background.

The result is a binary image with black pixels representing the background and white pixels representing the foreground.

Binary Thresholding: In a binary image, each pixel can only have two possible values: black (0) or white (1). Binary thresholding works by selecting a threshold value, which is a value that separates the darker and lighter pixel intensities in the grayscale image. Pixels with intensity values above the threshold are set to white, while those below the threshold are set to black. The selection of the threshold value has a significant impact on the accuracy of the segmentation and subsequent analysis. In Otsu Thresholding, the value of the threshold isn't chosen but is automatically determined. An image with two distinct image values is considered. The generated histogram of this image contains two peaks. Hence generally a threshold value that lies in the middle of both the histogram peak values is chosen. For this the traditional `cv2.threshold()` function with `cv2.THRESH_OTSU` as an extra flag is used. The syntax of `cv2.threshold()` function in OpenCV is as follows:

`cv2.threshold(source, threshold value, maxVal, thresholdingTechnique)` where the source is the input image in grayscale, the threshold value is the value of threshold below and above which pixel values will change accordingly, maxVal is the largest value that can be taken by a pixel, thresholding Technique: the type of thresholding algorithm to be applied.

Contouring

Contour detection is a task in image processing and computer vision that involves identifying and extracting the boundaries of objects in an image. A contour can be defined as the line joining all the points along the edge of an image having the same intensity. These boundaries are represented by curves or lines that connect points of similar color or intensity in the image. There are several techniques for contour detection in image processing, including:

- **Edge detection:** This involves detecting the edges in an image by looking for abrupt changes in intensity or color. These include the Sobel, Canny, and Laplacian filters.
- **Gradient-based methods:** These methods involve calculating the gradient of the image to identify the regions with the highest changes in intensity or color. These include the Prewitt and Roberts filters.
- **Thresholding:** This involves setting a threshold value and classifying pixels based on their intensity or color value. The resulting binary image is used to extract contours.
- **Region-based segmentation:** This involves dividing the image into regions based on color or texture similarity and identifying the boundaries between these regions.

Edge detection just gives the points in the image where the image intensity changes drastically. This may or may not form a closed shape. Whereas in contour detection, a closed shape (here hand gesture) is found and the boundary of the hand is detected. Only binary images resulting from the Otsu thresholding algorithm are passed to the `cv2.findContour()` method of OpenCV. This function takes 3 required arguments and 3 optional arguments. In this model, only the required parameters are used. The first is the binary image, the second is the contour retrieval mode. By using `cv2.RETR_TREE` retrieves all the contours according to their hierarchical relationship and in the order in which the contours are detected. The last argument to this function is the contour approximation method. In our case, we used `cv2.CHAIN_APPROX_NONE`, which stores all boundary points of

the hand. The function then returns a tuple with two elements, the first element is the contours detected on the image, and the second element is the hierarchy of the contours.

C. Convolutional Neural Network

A CNN is a type of deep learning algorithm mostly used for image recognition and processing tasks. A CNN is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. We'll be using a CNN model to extract features from the input frames and to predict hand gestures.

In our project, there are three convolutional layers with 16, 32, and 64 filters respectively. Next, the model has three pooling layers with (2x2), (3x3), and (5x5) pooling sizes. Lastly, it has two dense (fully connected) layers where one layer has ReLU activation and the other has Softmax activation functions.

D. Conversion

Finally, the class label obtained from the Fully Connected layer of CNN is converted to voice output using Pyttsx3 text-to-speech conversion library of Python and given as the final output.

IV. RESULTS

This section constitutes the snapshots of our system's user interface and outputs of a few signs performed. The first figure 4.1 shows the user interface, when clicking on the 'Start Camera' button the camera switches on and then two windows open.

The first one is divided into the region of interest where the sign performed in front of the camera appears and the second half is a blackboard with the predicted text printed. The second window displays the thresholded sign and the black-and-white image that will be processed further.

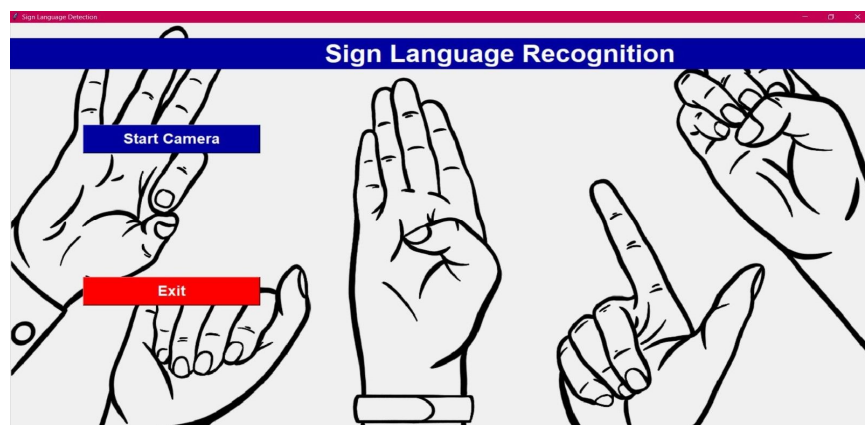


Fig: 4.1 GUI

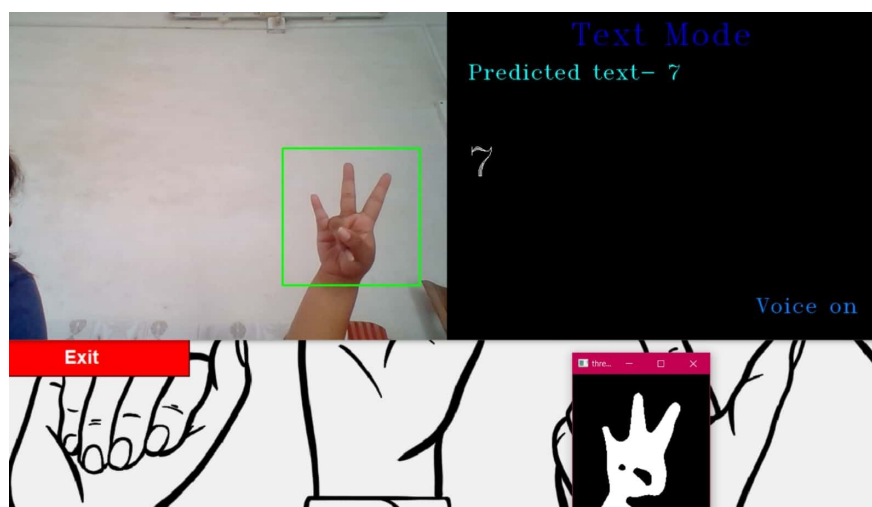


Fig: 4.2 Number – 7

The above figure 4.2 depicts the sign performed by the user for digit 7 and the text prediction on the right side of the first window.

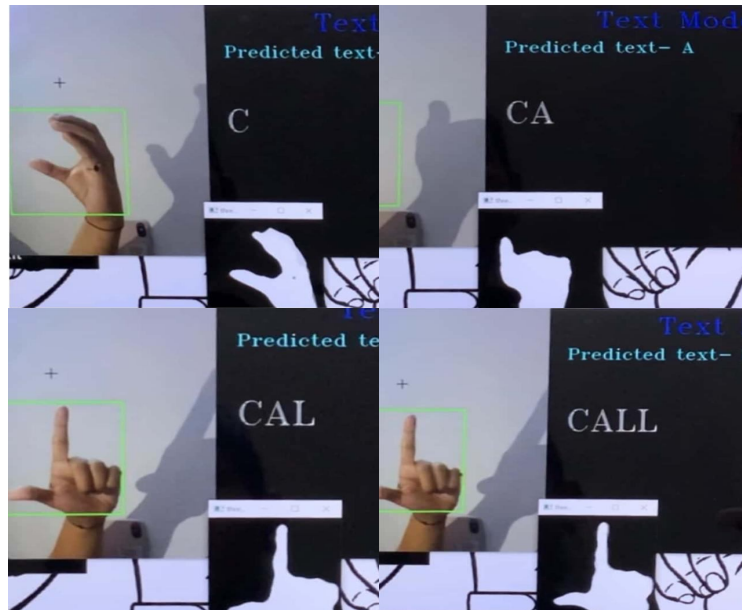


Fig: 4.3 Word – CALL

The above figure 4.3 depicts the signs performed by the user for the word ‘CALL’ and the text prediction on the right side of the first window.

V. CONCLUSION

Sign Language is used to bridge the communication gap between deaf-mute people and normal person. Sign language recognition using hand gestures is a difficult problem and our current work is only a small contribution towards achieving the results needed in the field of sign language gesture recognition. This system which is proposed above aims to do the same so as the two-way communication is possible.

The project in this paper is a step towards helping specially challenged people. The method proposed here facilitates the conversion of the sign into text and speech which overcomes the requirement of a translator since real time conversion is used. The system acts as voice of the person who is deaf-mute. This can be further enhanced by making it more user friendly, efficient, portable and compatible for more signs and as well as dynamic signs.

REFERENCES

- [1] Q. M. Areeb, Maryam, M. Nadeem, R. Alroobaea and F. Anwer, "Helping Hearing-Impaired in Emergency Situations: A Deep Learning-Based Approach," 2022.
- [2] Disha Shah, Aditi Pande, Yamini Salvi, Priti Gaddam, Prof. Mrs. S. A. Deshpande. "Sign Language Recognition for Deaf and Dumb".
- [3] Dhulipala, S.; Adedoyin, F.F.; Bruno, A. Sign and Human Action Detection Using Deep Learning, 2022.
- [4] Do, N., Kim, S., Yang, H., Lee, G.: Robust hand shape features for dynamic hand gesture recognition using multi-level feature LSTM, 2020.
- [5] A. Rao, P. V. V. Kishore, A. S. C. S. Sastry, D. A. Kumar, and K. Kumar.: Selfie continuous sign language recognition with neural network classifier', 2018.
- [6] Nagesh Kumar D N, Madhukar M, Adarsh Prabhakara, Archana V Marathe, Manoj, Shyam S Bharadwaj: "Sign Language to Speech Conversion – An Assistive System for Speech Impaired", 2019.