International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Sign Language Interpreter using Convolution Neural Network

Ghali Upendra[1], Kokkiligadda Bhuvanendra [2], D Gayathri[3]

[1]Student, Dept. Of CSE, SCSVMV (Deemed to be University), Kanchipuram, TamilNadu, India
[2]Student, Dept. Of CSE, SCSVMV (Deemed to be University), Kanchipuram, TamilNadu, India
[3]Assistant Professor, Dept. Of CSE, SCSVMV (Deemed to be University), Kanchipuram, TamilNadu, India

---

Abstract - *Sign language is one of the oldest and most natural forms of language for communication. These languages are basically used to aid deaf people but since most people do not know sign language and interpreters are very difficult to come by we have come up with a real time method using neural networks for hand gestures based American Sign Language. In our method, the hand gesture is first passed through a camera and it is preprocessed and is passed through an interpreter which predicts the hand gestures.*

***Key Words*: Convolution Neural Network, Thresholding, Gaussian Blur, Pre-processing**

1. INTRODUCTION

In order to overcome the gap in communication caused by the difference in modes of communication, an interpreter is necessary to reduce the confusion. This project is an attempt to ease the communication between deaf and normal people. The main objective of the project is to translate sign language to text language. In this project, the recognition of hand gestures is done with the usage of Convolution Neural Network. The gesture made by the hand inside the recognition area in each frame is matched with the pre-classified data available in the dataset and when a match is declared, the corresponding alphabet/number is given as a text and speech output that is easily understandable by a normal person who doesn't know the sign language.

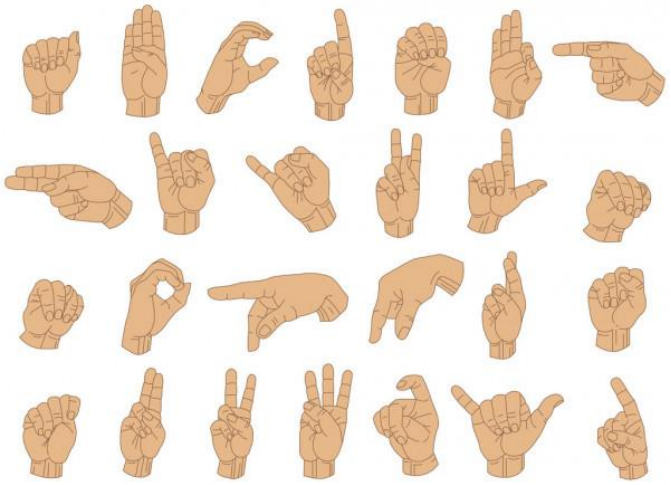
1.1 LITERATURE SURVEY

In the recent years there has been tremendous research done on the sign language recognition. After thorough checking of literature survey we noticed the basic steps in hand gesture recognition are:

● Data Acquisition ● Data Pre-processing ● Training ● Gesture classification

Data Acquisition:

It is a process where we collect the data i.e., hand gestures for all the alphabets through webcam and divide every gesture with respect to the corresponding alphabet. The main challenge of this is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin-color possibilities as well as differences in view points, scales, and speed of the camera while capturing the scene.



Data Pre-processing:

However the pictures captured are colored images, so it takes more processing time. To reduce this, we have to convert those into black and white and have to remove all the lines through thresholding and Gaussian blur.

Training:

After that we have to train the dataset to recognize the gestures. It is done through Convolution Neural Networks which is defined through several layers.

Prediction:

It requires a Graphical User Interface (GUI) which has to user friendly. Once provided with a gesture through webcam, it has to check with the gestures in the trained dataset and produce an accurate output in the form of text.

2. Convolution Neural Network

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. And also, the final output layer will have dimensions (number of classes), because by the end of the CNN architecture we can reduce the full size image into a single vector of class scores.

A. Convolution Layer:

In convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration, we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color.

B. Pooling Layer:

We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type of pooling:

a) Max Pooling: In max pooling we take a window size [for example window of size 2*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.

b) Average Pooling: In average pooling we take average of all values in a window.

C. Fully Connected Layer:

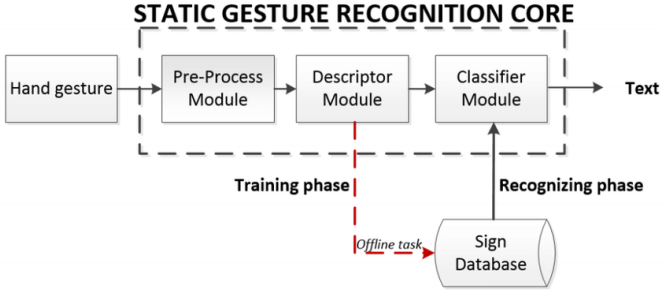
In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

D. Final Output Layer:

After getting values from fully connected layer, well connect them to final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

3. Methodology

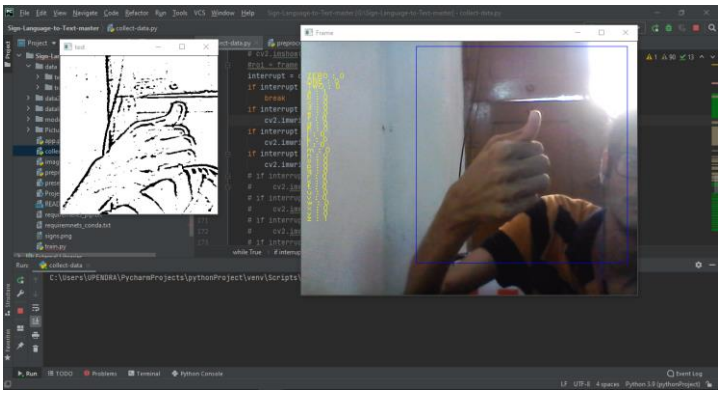
The system is a vision based approach. Most of the signs are represented with bare hands to eliminate the problem of using any artificial devices for interaction.



Data Set Generation:

For the project we tried to find already made datasets but we couldn't find dataset in the form of raw images that matched our requirements. There are only datasets in the form of RGB values. Hence we decided to create our own data set. Steps we followed to create our data set are as follows.

We used Open computer vision (OpenCV) library in order to produce our dataset. At first we captured around 1000 images of each of the symbol in ASL to train data and around 200 images per symbol for testing purpose. First we capture each frame shown by the webcam of our machine. In the each frame we define a region of interest (ROI) and apply RGB and then thresholding and Gaussian blur as shown in the image below:



3.1 ALGORITHM

Algorithm Layer 1:

1. Apply Gaussian blur filter and threshold to the frame taken with opencv to get the processed image after feature extraction.

2. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word.

3. Space between the words are considered using the blank symbol.

Algorithm Layer 2:

1. We detect various sets of symbols which show similar results on getting detected.

2. We then classify between those sets using classifiers made for those sets only.

3.2 CNN MODEL

1st Convolution Layer:

The input picture has resolution of 128x128 pixels. It is first processed in the first convolution layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weight.

1st Pooling Layer:

The pictures are down sampled using max pooling of 2x2 i.e. we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.

2nd Convolution Layer:

Now, this 63 x 63 from the output of the first pooling layer is served as an input to the second convolution layer. It is processed in the second convolution layer using 32 filter weights (3x3 pixels each).This will result in a 60 x 60 pixel image.

2nd Pooling Layer:

The resulting images are down sampled again using max pool of 2x2 and are reduced to 30 x 30 resolutions of images.

1st Densely Connected Layer:

Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolution layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of this layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid over fitting.

2nd Densely Connected Layer:

Now the output from the 1st Densely Connected Layer is used as an input to a fully connected layer with 96 neurons.

Final layer:

The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

4. CHALLENGES FACED

There were many challenges faced by us during the project. The very first issue we faced was of dataset. We wanted to deal with raw images and those too square images as CNN in Keras as it was a lot more convenient working with only square images. We couldn't find any existing dataset for that hence we decided to make our own dataset. Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provide that image as input for CNN model. We tried various filter including binary threshold, canny edge detection, Gaussian blur etc. but finally we settled with Gaussian blur filter. More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

5. RESULTS

We have achieved an accuracy of 95.8% in our model using only layer 1 of our algorithm, which is a better accuracy then most of the current research papers on American Sign Language. Most of the research papers focus on using devices like kinect for hand detection.

A sensor like kinect not only isn't readily available but also is expensive for most of audience to buy and our model uses a normal webcam of the laptop hence it is great plus point and also led cost efficient with more accuracy rate.

6. CONCLUSION

In this report, a functional real time vision based American Sign Language recognition for D&M people have been developed for ASL alphabets. We achieved final accuracy of 98.0% on our dataset. We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other. This way we are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

REFERENCES

[1] T. Yang, Y. Xu, and "A. , Hidden Markov Model for Gesture Recognition", CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ.,Pittsburgh,PA, May 1994..

[2] Pujan Ziaie, Thomas M¨uller , Mary Ellen Foster , and Alois Knoll"A Na¨ıve Bayes Munich,Dept. of Informatics VI, Robotics and Embedded Systems,Boltzmannstr. 3, DE-85748 Garching, Germany.

[3] Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.

[4] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham.

BIOGRAPHIES

Ghali Upendra is pursuing B. Eng. from SCSVMV (Deemed to be University).

Kokkiligadda Bhuvanendra is pursuing B. Eng. from SCSVMV (Deemed to be University).

Ms. D. Gayathri is Assistant Professor in Computer science and Engineering department in SCSVMV (Deemed to be University).