Sign Language Recognition with Convolution Neural Network

Sharath Kumar D A

Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore, Karnataka, INDIA

---***---

ABSTRCT: Sign language is used by deaf and hard hearing people to exchange information between their own community and with other people. Computer recognition of sign language deals from sign gesture acquisition and continues till text/speech generation. Sign gestures can be classified as static and dynamic. However static gesture recognition is simpler than dynamic gesture recognition but both recognition systems are important to the human community. The sign language recognition steps are described in this project. The data acquisition, data pre-processing and transformation, feature extraction, classification and results obtained are examined. We also discuss the challenges and limitations faced by sign language recognition. Overall, it is hoped that the project may provide readers with a comprehensive introduction into the field of sign language recognition, and further facilitate future research efforts in this area.

1. INTRODUCTION

The Sign Language development is different for each country or sub-continent. The following table 1 presents the development of sign languages of influencing countries/sub-continent. In 1880, the National Association of the Deaf (NAD) was established by deaf leaders for the right of the American deaf community to use sign language. The NAD struggles to recognize the sign language from birth is a human right for every person, and that deaf infants and children should be given the opportunity to acquire and develop proficiency in ASL as early as possible. This position is also in line with the stance of the World Federation of the Deaf and the United Nations on human rights, including the recognition of sign languages.

Sl.No	Country/Subcontinent	Sign Language	Abbreviation
1	United Kingdom	British Sign Language	BSL
2	United States of America	American Sign Language	ASL
3	Japan	Japanese Sign Language	JSL
4	Peoples Republic of China	Chinese Sign Language	CSL
5	Republic of India	Indian Sign Language	ISL

Table 1. Sign Language development is different for each country or sub-continent

A good Sign Language Recognition (SLR) system can overcomes the barriers exists between the speech and hearing people with speaking society. The goal of SLR for developing systems and approaches for properly recognize the series of gestures and to know the meaning of the gestures Helen Cooper, Brian Holt, Richard Bowden, "Sign Language Recognition". Several methodologies to SLR imperfectly treat the difficulty as Gesture Recognition (GR). SL is a challenge that it is multi-channel; translating meaning via several manners at the same time. As the research of SL semantics are yet in its initial junctures, it is necessary to develop a novel, universal SLR system. SLR is a challenging and motivational task with respect to many constraints and factors. Some of the motivational factors for opting up the design of ASL Recognition System are discussed here. SLR is a noticeable task due to its impact on humanoid society as the mute pupil facing huge communication gap with the speaking community.

SLR is a challenging work because of its variation of hand gestures, facial expressions, body movements and many such variations and confines in this regard. A very less volume of work has been done in this lane to recognize the distance invariant, size invariant, rotation invariant, and race invariant ASL gestures with respect to background (plain and complex, uniform and non-uniform), location (indoor and outdoor), time (day and night), and light illumination (natural and artificial). Also, there is no noticeable work has been done predominantly for real time gesture recognition by considering various research disputes with respect to static and dynamic environment. There are huge amount of opportunities to carry out the research in recognizing ASL to make the communication easier between and mute and speaking community.

The objective of this project is to accomplish the transliteration of 24 static sign language alphabets of American Sign Language into humanoid or machine decipherable English manuscript.

- Pre-processing operations of the signed input gesture are done in the first phase like resizing and converting to gray-scale.
- In the next phase, the various region properties of pre-processed gesture image is computed by using methods like Canny edge detection.
- In the final phase, based on the properties calculated of earlier phase, the transliteration of signed gesture into text has been carried out by a Convolution Model.

2. METHODOLGOY

The proposed method as follows

2.1. Algorithm

Step 1: **Image Acquisition:** Read the RGB image from the data set

Step 2: **Image Segmentation:** Convert the image to Gray-scale, later extracting edge from the image using canny edge detection method.

Step 3: **Image pre-possessing:** The output image from canny edge detection is resized to a definite size (200*200).

Step 4: **Class Labelling:** The images are labelled using one hot encoder i.e [1,0,0],[0,1,0],[0,0,1] for a,b,c as an example.

Step 5: **Mathematical representation:** Convert the image to matrix format with one channel.

Step 6: **Model:** Prepare a CNN model with convolution, max-pooling, dense layers with an activation function. Later fit the image matrix into CNN Model.

2.2. Design of Model

We have done the project on a Convolution Neural Network Model which is designed with 4 continuous convolution + relu + max-pooling modules. Our convolutions contain a filter of size 3x3 and our max-pooling has a pooling layer of 2x2. The max-polling reduces the feature map by half but convolution does not affect the size of feature map as we are not using an stride. Relu(Rectified Linear Unit) is an activation function used in the convolution layer which help the network use the important information and suppress the irrelevant data points. In the end we have 2 dense layers in our model which are fully connected network. The one with a fully connected layer with 500 neurons and relu as the activation function and other one dense layer is used to classify the input image to 28 different classes i.e A to Z, del and space. For this purpose we use **Softmax** function as it a multi class problem. In our model we use categorical-crossentropy loss function as we have 28 different classes. We use Adadelta optimizer in the end fully connected layer which is an derived optimizer from Gradient descent. This is used to restricts the window of accumulated past gradients to some fixed size.

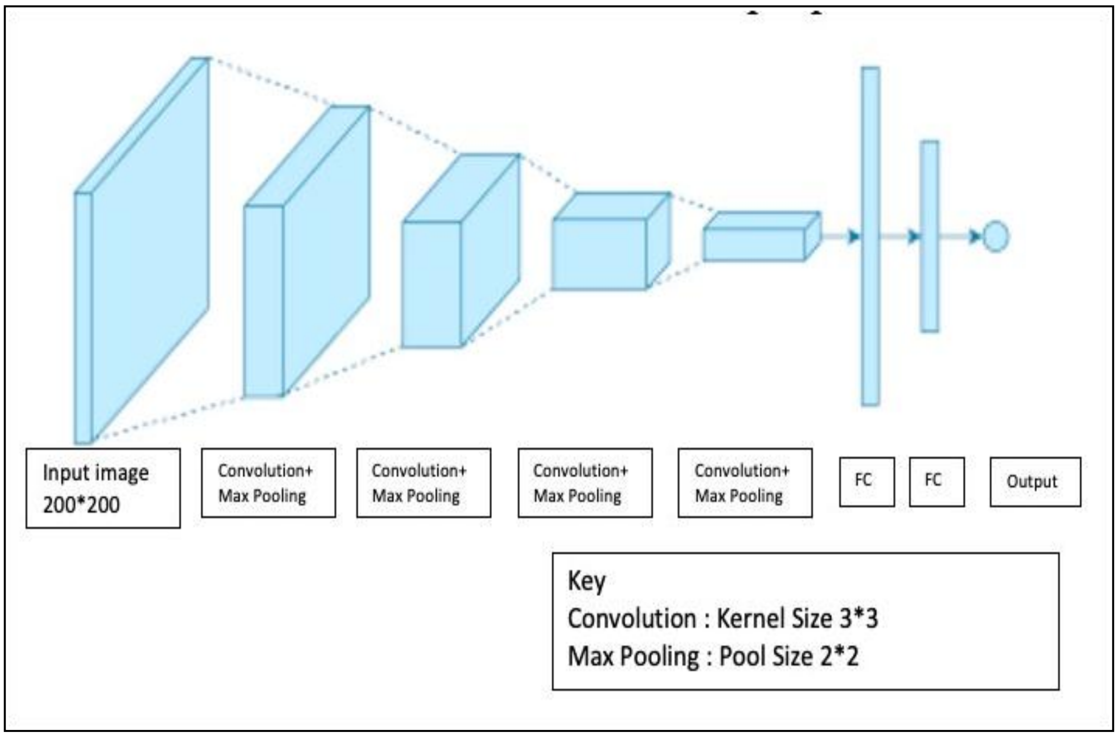


Figure 1. Model

2.3. Mathematical Equations

Convolution key step in CNN model for extracting features. It is a process where we take a small matrix of numbers (called kernel or filter), we pass it over our image and transform it based on the values from filter. Subsequent feature map values are calculated according to the following formula, where the input image is denoted by f and our kernel by h. The indexes of rows and columns of the result matrix are marked with m and n respectively.

$G[m,n] = (f * h)[m,n] = {}^{XX}h[j,k]f[m - j,n - k]$

$j\ k$

After placing our filter over a selected pixel, we take each value from kernel and multiply them in pairs with corresponding values from the image. Finally we sum up everything and put the result in the right place in the output feature map. Above we can see how such an operation looks like in micro scale, but what is even more interesting, is what we can achieve by performing it on a full image. Figure 4 shows the results of the convolution with several different filters.

The number of output features after passing through a kernal is

$$n_{out} = [\frac{n_{in} + 2p - k}{s} + 1]$$

Where n_{in} is number of input features, n_{out} is the number of output features, k is the convolution kernel size, p convolution padding size, s convolution stride size.

Max Pooling is very often after convolution layers in CNN Model. They are used primarily to reduce the size of the tensor and speed up calculations. This layers are simple we need to divide our image into different regions, and then select a maximum value from each region and put it in the corresponding place in the output. We have two hyper-parameters available filter-size and stride. If you are performing pooling for a multi-channel image, the pooling for each channel should be done separately.

Adadelta Optimizer is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w. Instead of inefficiently storing w previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average $E[g^2]_t$ at time step t then depends only on the previous average and the current gradient:

$E[g^2]_t = \gamma[g^2]_{t-1} + (1 - \gamma)g_t^2$

The root mean squared error of parameter updates is thus:

$$RMS[\Delta\theta]_t = \sqrt{(E[\Delta\theta^2]_t + \epsilon)}$$

Since $RMS[\Delta\theta]_t$ is unknown, we approximate it with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\Delta\theta]_{t-1}$ finally yields the Adadelta update rule:

$$\Delta\theta = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t}$$
$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

2.4. Experimental Analysis

2.4.1 Dataset

In our experiment we use the American Sign Language dataset, which contains 28 different classes i.e A to Z , Space ,Del as the Classes. The dataset consists of 84,000 images of size 200*200 with each Class having 3000 images captured from different angles, different ways of expressing a sign and at different distance from camera.

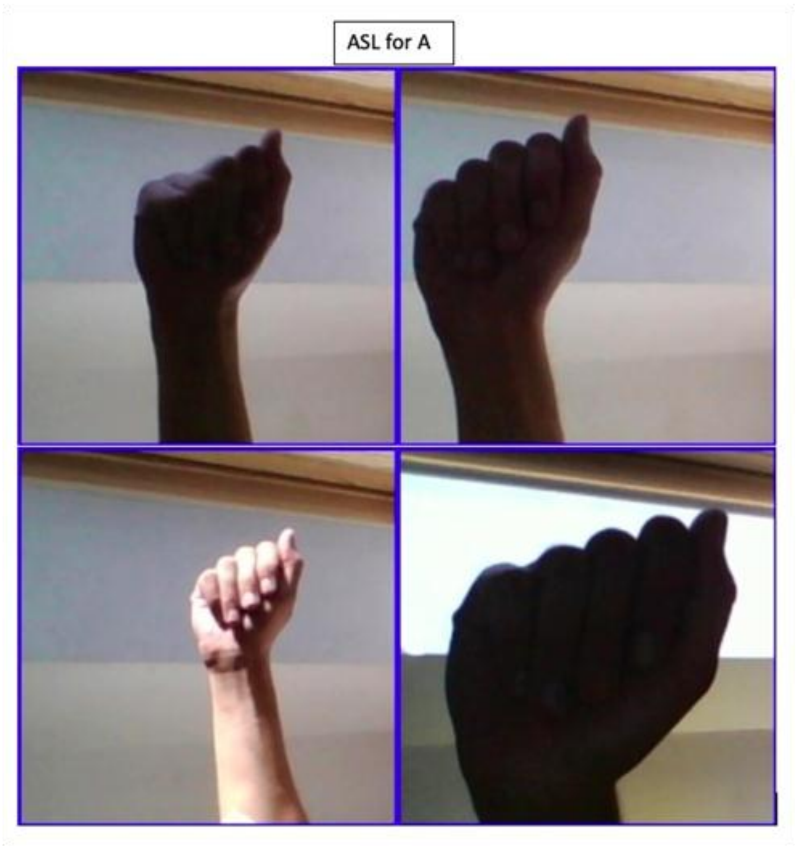
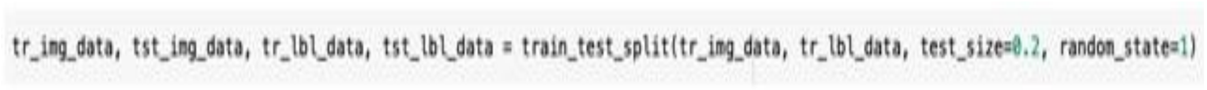


Figure 2. American Sign Language dataset model

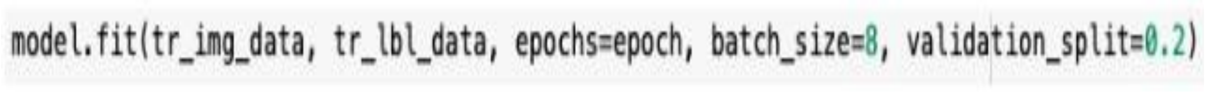
The above image is some examples of ASL of 'A' which is present in the dataset. The other alphabets,space and Del also consist of such similar images with their signs in the respective folders with images and Class as the folder name.

2.4.2 Experimental Settings

In our experiment we split the dataset into 3 part training, testing and validation dataset with a ratio of 6:2:2 respectively. We use Keras to build the CNN model, With Api's of Layers and models in Keras.We have set dropout of 0.3 and 0.4. We have set batch size as 8 which means while training the model takes 8 images as input at a time. Epoch is the number of times the model is trained on the same dataset. This may improve or decrease the accuracy of the model as large epoch can lead to over-fitting of the model or under-fitting of model.In our project we have done experiment on two epochs i.e 5 and 10.



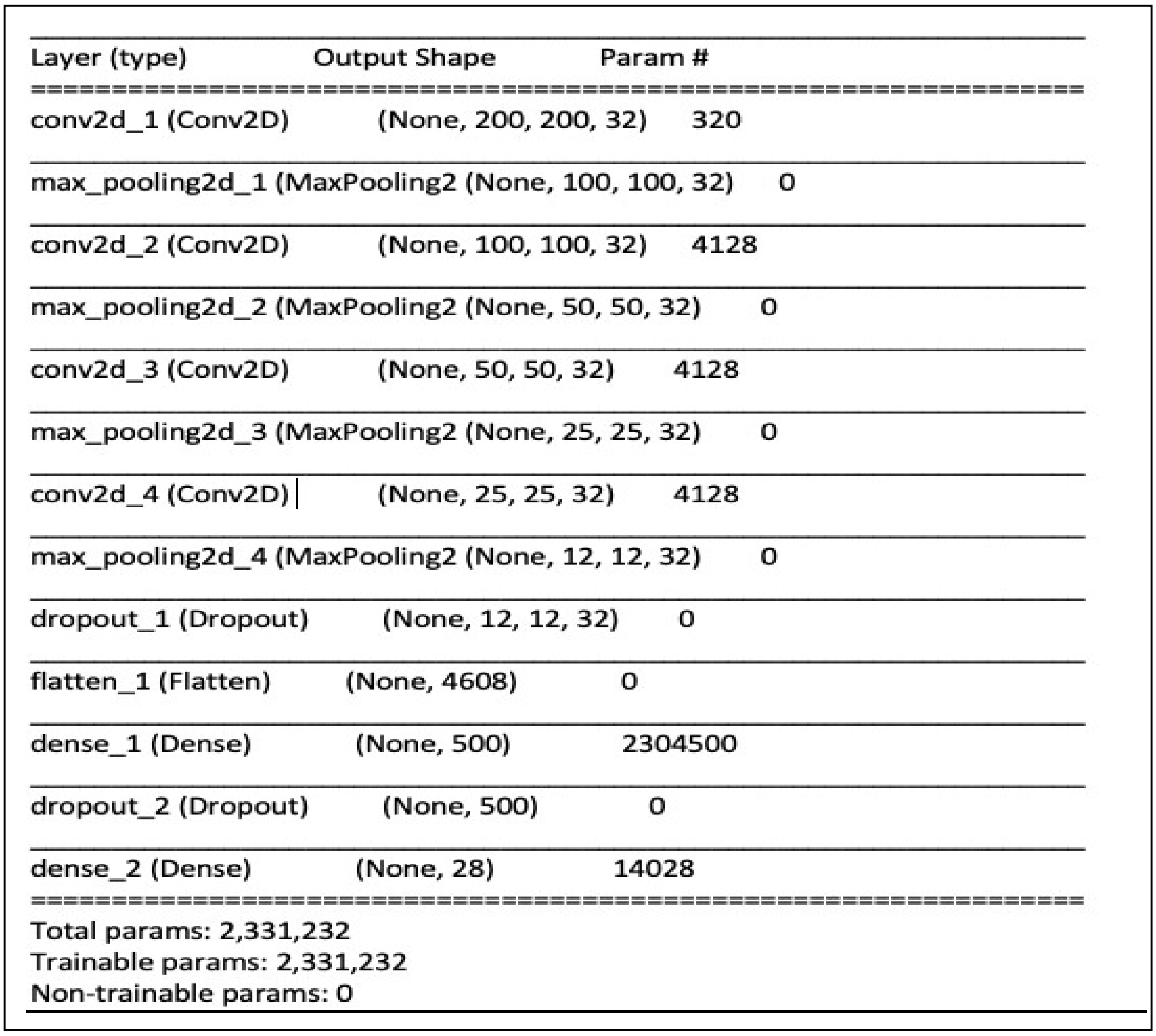
```
tr_img_data, tst_img_data, tr_lbl_data, tst_lbl_data = train_test_split(tr_img_data, tr_lbl_data, test_size=0.2, random_state=1)
```



```
model.fit(tr_img_data, tr_lbl_data, epochs=epoch, batch_size=8, validation_split=0.2)
```

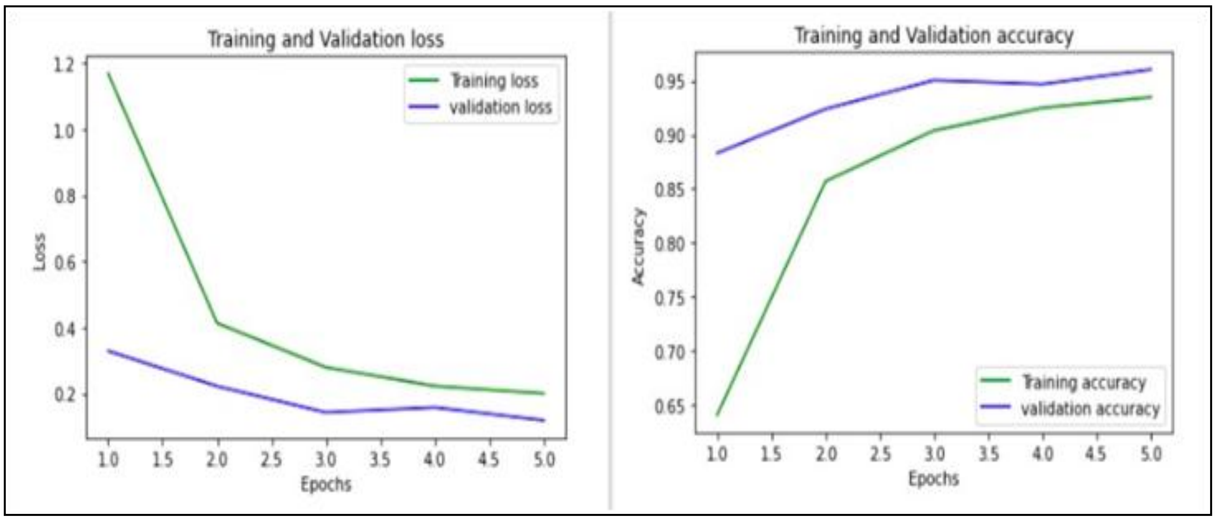
3. RESULTS

Experiment is carried out for 2 different runs on training images: 1st run for 5 epoch and 2nd run fro 10 epoch. The model structure used for training is displayed bellow:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 32)      320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 100, 100, 32)      4128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 50, 50, 32)        4128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 25, 25, 32)        4128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               2304500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 28)                14028
=================================================================
Total params: 2,331,232
Trainable params: 2,331,232
Non-trainable params: 0
_________________________________________________________________
```

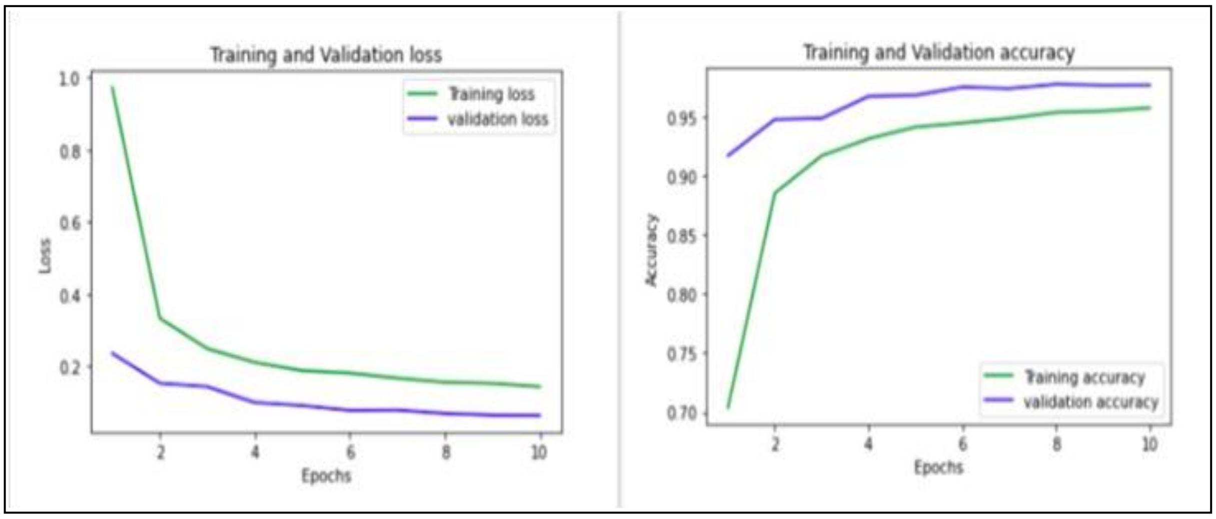
un1: epoch=5 The Validation loss ,Training loss ,Validation Accuracy and Training Accuracy is plotted on a graph to check if the model is not over fitted.



The results for testing on 16800 images are as follows:

Testing Loss	Testing Accuracy
0.12415	0.96172

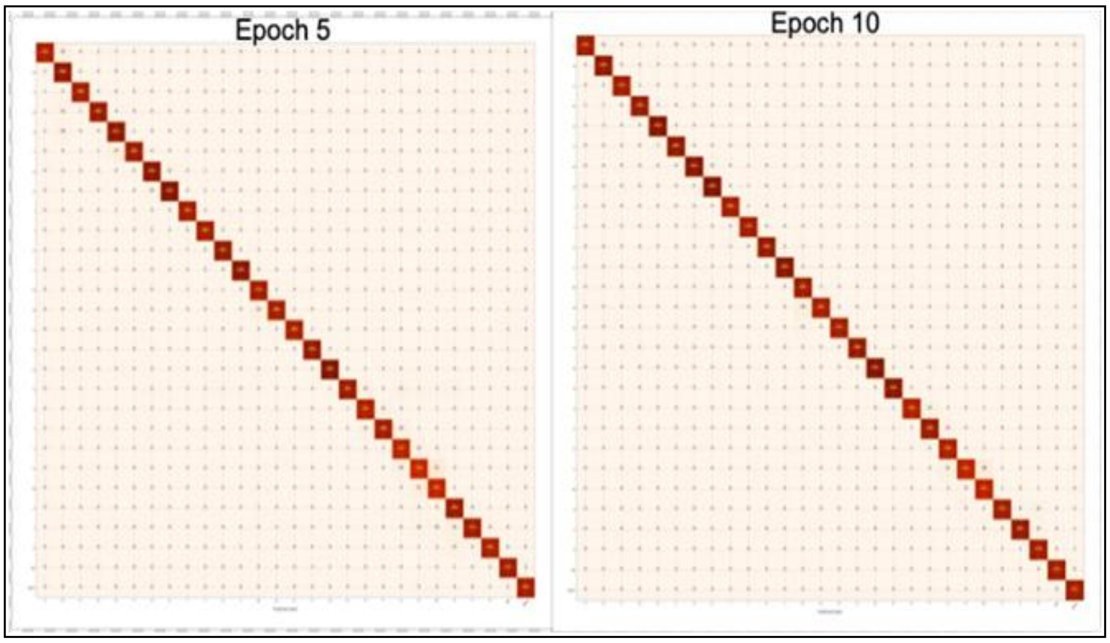
Run2: epoch=10 The Validation loss ,Training loss ,Validation Accuracy and Training Accuracy is plotted on a graph to check if the model is not over-fitted.



The results for testing on 16800 images are as follows:

Testing Loss	Testing Accuracy
0.06615	0.97750

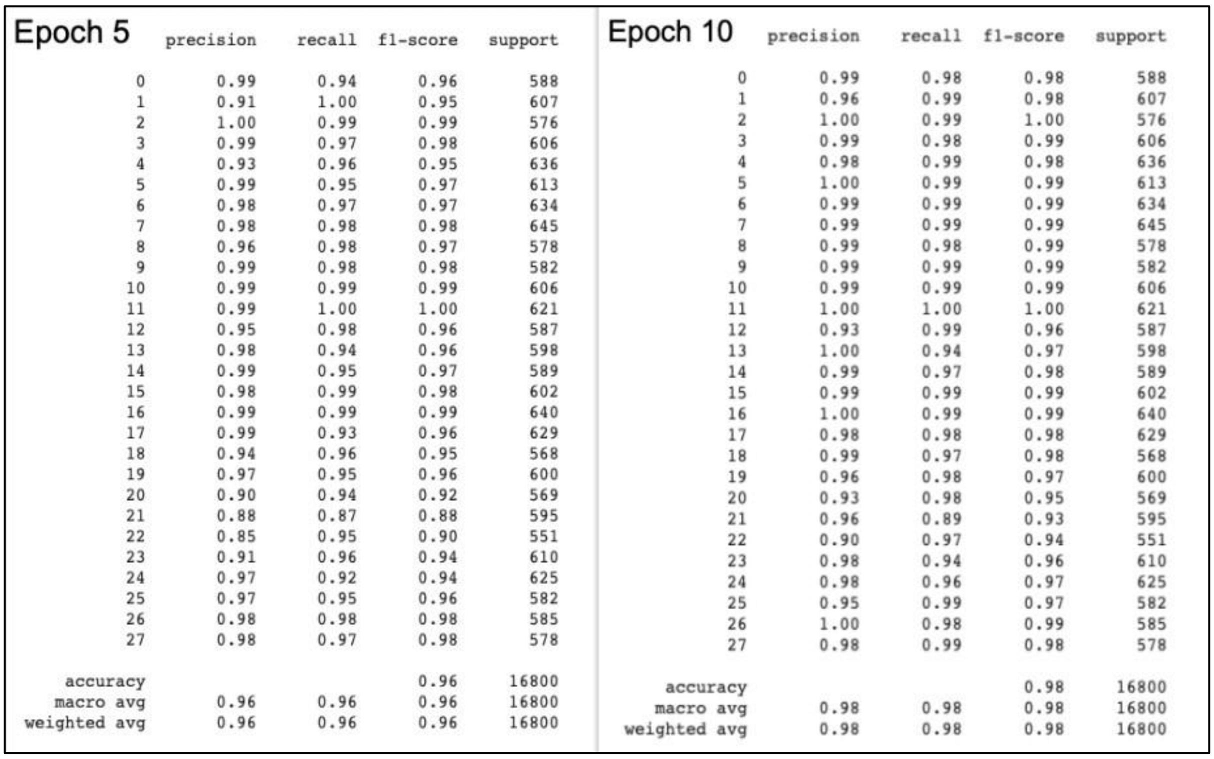
The bellow image is a Confusion Matrix which represents the number of Images that are predicted accurately and the number of images that and miss-predicted with other classes. The Diagonal in the image represents the number of images that are correctly predicted of that particular class represented in the row or column.



To get a clear insight of the Confusion Matrix we look at the precision and recall. Bellow is the image of precision and recall values for every class respectively where 0-25 represents A-Z in order, 26 represents del and 27 represents space. These values are obtained from the same testing set images.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

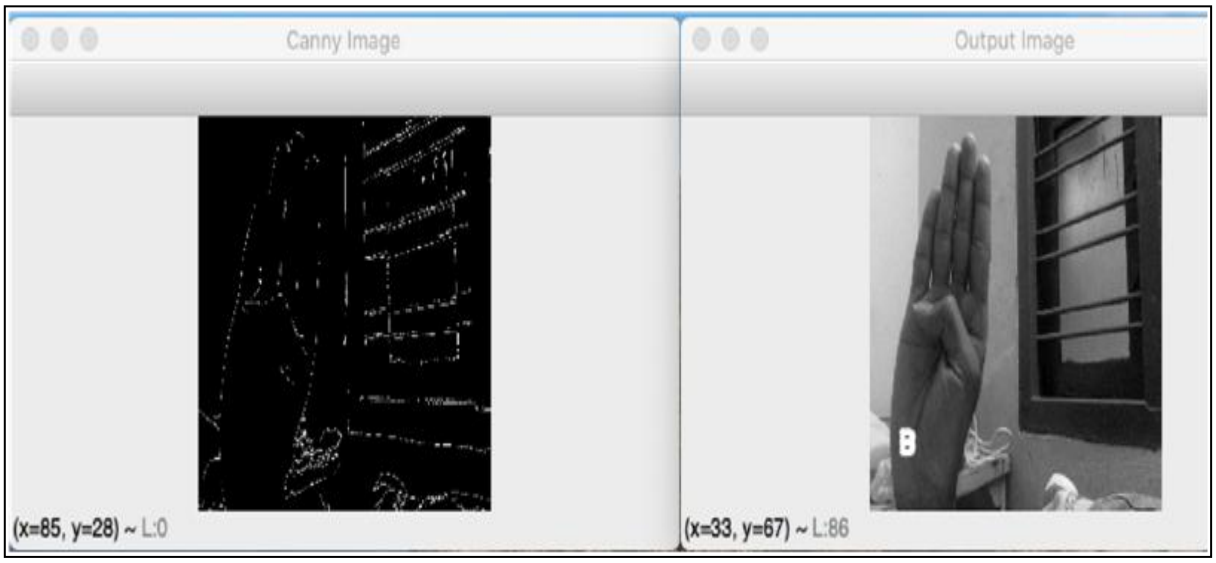
$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$



```
Epoch 5       precision    recall  f1-score   support

           0       0.99      0.94      0.96       588
           1       0.91      1.00      0.95       607
           2       1.00      0.99      0.99       576
           3       0.99      0.97      0.98       606
           4       0.93      0.96      0.95       636
           5       0.99      0.95      0.97       613
           6       0.98      0.97      0.97       634
           7       0.98      0.98      0.98       645
           8       0.96      0.98      0.97       578
           9       0.99      0.98      0.98       582
          10       0.99      0.99      0.99       606
          11       0.99      1.00      1.00       621
          12       0.95      0.98      0.96       587
          13       0.98      0.94      0.96       598
          14       0.99      0.95      0.97       589
          15       0.98      0.99      0.98       602
          16       0.99      0.99      0.99       640
          17       0.99      0.93      0.96       629
          18       0.94      0.96      0.95       568
          19       0.97      0.95      0.96       600
          20       0.90      0.94      0.92       569
          21       0.88      0.87      0.88       595
          22       0.85      0.95      0.90       551
          23       0.91      0.96      0.94       610
          24       0.97      0.92      0.94       625
          25       0.97      0.95      0.96       582
          26       0.98      0.98      0.98       585
          27       0.98      0.97      0.98       578

    accuracy                           0.96     16800
   macro avg       0.96      0.96      0.96     16800
weighted avg       0.96      0.96      0.96     16800
```

```
Epoch 10      precision    recall  f1-score   support

           0       0.99      0.98      0.98       588
           1       0.96      0.99      0.98       607
           2       1.00      0.99      1.00       576
           3       0.99      0.98      0.99       606
           4       0.98      0.99      0.98       636
           5       1.00      0.99      0.99       613
           6       0.99      0.99      0.99       634
           7       0.99      0.99      0.99       645
           8       0.99      0.98      0.99       578
           9       0.99      0.99      0.99       582
          10       0.99      0.99      0.99       606
          11       1.00      1.00      1.00       621
          12       0.93      0.99      0.96       587
          13       1.00      0.94      0.97       598
          14       0.99      0.97      0.98       589
          15       0.99      0.99      0.99       602
          16       1.00      0.99      0.99       640
          17       0.98      0.98      0.98       629
          18       0.99      0.97      0.98       568
          19       0.96      0.98      0.97       600
          20       0.93      0.98      0.95       569
          21       0.96      0.89      0.93       595
          22       0.90      0.97      0.94       551
          23       0.98      0.94      0.96       610
          24       0.98      0.96      0.97       625
          25       0.95      0.99      0.97       582
          26       1.00      0.98      0.99       585
          27       0.98      0.99      0.98       578

    accuracy                           0.98     16800
   macro avg       0.98      0.98      0.98     16800
weighted avg       0.98      0.98      0.98     16800
```

The bellow image contains the image after canny edge detection is carried out on the input image and the right side of the image is the output with the prediction of the class. This image is captured live which is not contained in the dataset which we have used for training. This image has comparatively more noise in the background.



4. DISCUSSION AND CONCLUSION

The accuracy of the model on the whole is good with above 96% but if we look at individual letter prediction accuracy we can see that some letter like E,S,U,V,W,X have less accuracy at epoch 5 but these values are increased at epoch 10 to a high accuracy comparatively only U,W have less precision when compared to other letters in epoch 10.By viewing the graph of training also we can see that model is not over fitted and both validation and training accuracy are close enough which shows that model is well trained.

Time complexity for individual Convolution layer is $channeloutput * (outputimagesize) * (filtersize * channelinput)$

The time complexity for a CNN model with 4 layers and an image of Matrix Size $\boldsymbol{M_{ij}}$ □ $\boldsymbol{M_{jk}}$,**l** nodes and with t training examples and n epochs.

The result is $\boldsymbol{\Theta(nt}$ □ $\boldsymbol{(ij + jk + kl))}$

The results are good over the dataset given but for some letters there is less precision due to the similarity with other signs like ’U’ and ’V’ have similar hand sign with just a small gap in-between the fingers which lead to ambiguity for the model to predict.The model is able to capture maximum features from the input image the prediction percentage value is also high when model is predicting a class. We can try to improve the dataset with more noise in the background which is more realistic to real world. Later we can develop an application which uses camera to read the video frame by frame and detect the sign and print the sentence with a proper meaning bellow which can be helpful to many people who would like to communicate with dumb and deaf people. We can also improvise in which on typing a sentence can produce a video with signs so that dumb and deaf people can understand what ordinary people are trying to communicate.

REFERENCES

[1] Srinath S and Ganesh Krishna Sharma. ”Classification approach for Sign Language Recognition”. International Conference on Signal, Image Processing, Communication Automation, 2017.

[2] Jayshree R. Pansare and Maya Ingle. ”Vision-Based Approach for American Sign Language Recognition Using Edge Orientation Histogram”. International Conference on Image, Vision and Computing, pp.86-90, 2016.

[3] Matheesha Fernando and Janaka Wijayanayaka. ”Low cost approach for Real Time Sign Language Recognition”. 8th International Conference on Industrial and Information Systems, 2013.

[4] Nagaraj N. Bhat, Y V Venkatesh, Ujjwal Karn, and Dhruva Vig. ”Hand Gesture Recognition using Self Organizing Map for Human Computer Interaction”. International Conference on Advances in Computing, Communications and Informatics, 2013.

[5] Fahad Ullah. ”American Sign Language Recognition System for Hearing Impaired People Using Cartesian Genetic Programming”. 5th International Conference on Automation, Robotics and Applications, 2011.

[6] Asha Thalange and Dr. S. K. Dixit. ”COHST and Wavelet Features Based Static ASL Numbers Recognition”. 2nd International Conference on Intelligent Computing, Communication Convergence (Elsevier), 2016.

[7] AshaThalange and ShantanuDixit. ”Effectofthinningextent on ASL number recognition using open-finger distance feature measurement technique”. International Conference on Signal Processing And Communication Engineering Systems, 2015.

[8] Sharmila Konwar, Sagarika Borah, and Dr.T Tuithung. ”Effectofthinningextent on ASL number recognition using open-finger distance feature measurement technique”. IEEE International Conference on Communication and Signal Processing, 2014.

[9] Sruthi Upendran and Thamizharasi. ”American Sign Language Interpreter System for Deaf and Dumb Individuals”. International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014.

[10] Aditi Kalsh and N.S. Garewal. ”Sign Language Recognition for Deaf Dumb”. International Journal of Advanced Research in Computer Science and Software Engineering, 2013.

[11] Sriparna Saha, Rimita Lahiri, Amit Konar, and Atulya K. Nagar. ”A Novel Approach To American Sign Language Recognition Using MAdaline Neural Network”. IEEE Symposium Series on Computational Intelligence (SSCI), 2016.

[12] Geetha M, Rohit Menon, Suranya Jayan, Raju James, and Janardhan G.V.V. ”Gesture Recognition for American Sign Language with Polygon Approximation”. IEEE International Conference on Technology for Education, 2011.

[13] Nachamai. M. ”Alphabet Recognition of American Sign Language: A Hand Gesture Recognition Approach Using Sift Algorithm”. International Journal of Artificial Intelligence Applications, 2013.

[14] Davi Hirafuji Neiva and Cleber Zanchettin. ”A Dynamic Gesture Recognition System to Translate Between Sign Languages in Complex Backgrounds”. 5th Brazilian Conference on Intelligent Systems, 2016.

[15] Singha.J and Das.K. ”Hand Gesture Recognition Based on KarhunenLoeve Transform”. Mobile and Embedded Technology International Conference, 2013.

[16] R. Sharma. ”Recognition of Single Handed Sign Language Gestures using Contour Tracing descriptor”. Proceedings of the World Congress on Engineering, 2013.

[17] Zhi-huaChen, Jung-TaeKim, JianningLiang, JingZhang, and Yu-Bo Yuan. ”Real-Time Hand Gesture Recognition Using Finger Segmentation”. Hindawi Publishing Corporation, the Scientific World Journal, 2014.

[18] Suchin Adhan and Chuchart Pintavirooj. ”Alphabetic Hand Sign Interpretation using Geometric Invariance”. IEEE International Conference on Biomedical Engineering, 2014.

[19] Taehwan Kim, Karen Livescu, and Gregory Shakhnarovich. ”American Sign Language Finger spelling Recognition with phonological featurebased tandem models”. IEEE workshop on Spoken Language Technology, 2012.