**Report**

**Sign Language Detection Using Image Processing and Convolutional Neural Networks**

**Team members:**

Nachammai Devi Pooja S (185001096)

Prashanth S (185001111)

Prathyush S (185001112)

RamaPrabha R (185001123)

# About the project

Sign language (SL) is the way of communication among the Deaf-Dumb people. SLs are gestural languages which uses sign message for communication by hand without speaking. There arises the need for sign language interpreters who can interpret sign language to spoken language and vice versa. But, the availability of such interpreters is limited, expensive and does not work throughout the life period of a deaf person. Automatic sign language recognition systems which could automatically translate the signs into corresponding text or voice without the help of sign language interpreters gives the chance to deaf people to express their idea without human translator. This project, Sign language Recognition approaches that aims to provide communication way for Deaf and Dumb Community over Society. It describes review of Image based sign language recognition system. Signs are in the form of hand gestures and these gestures are identified from images.

# Methodology

## Dataset

The data used for training and testing came from our own self-made dataset.120 samples of each of the 26 signed letters (a, b, c, d, e, f, g, h, i,...,x,y,z) and one empty sample considered as nothing were collected. Each sample consists of a person signing the corresponding letter. This dataset consists of 3,240 images of hand gesture. The following is a visualization of some of the unprocessed raw images in the dataset:



Fig 1 - (unprocessed raw image)

Image Processing:

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image.

In Sign Language Recognition Image processing is used to better extract features from input images. Images are in static image or dynamic image of sign perform by human. In particular, the features that we extract from hand gesture images should be invariant to background data, translation, scale, shape, rotation, angle, coordinates, movements etc.

The steps involved in the process are:

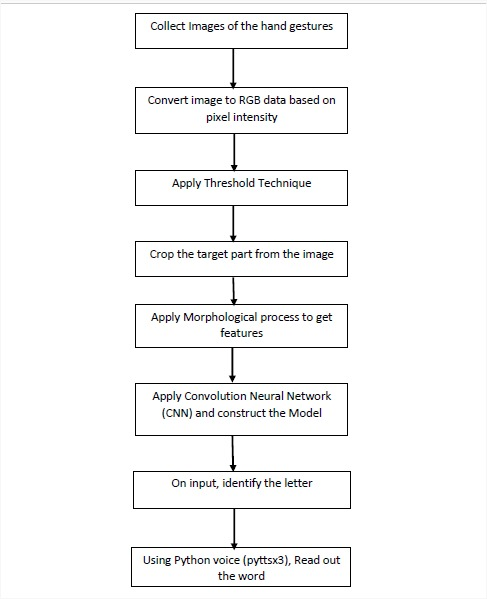
a. First, images are converted to RGB data from colored based on pixel intensity.

b. Since the background data is not tested, it should be remove the background from the foreground. By subtracting the background image from an input image.

c. Next step is apply threshold technique to ensure that hand pixels would not be subtracted out, and it will be converted to binary image.

d. Then crop the target part and evaluate various morphological features from image using various feature extraction algorithms.

System Architecture:



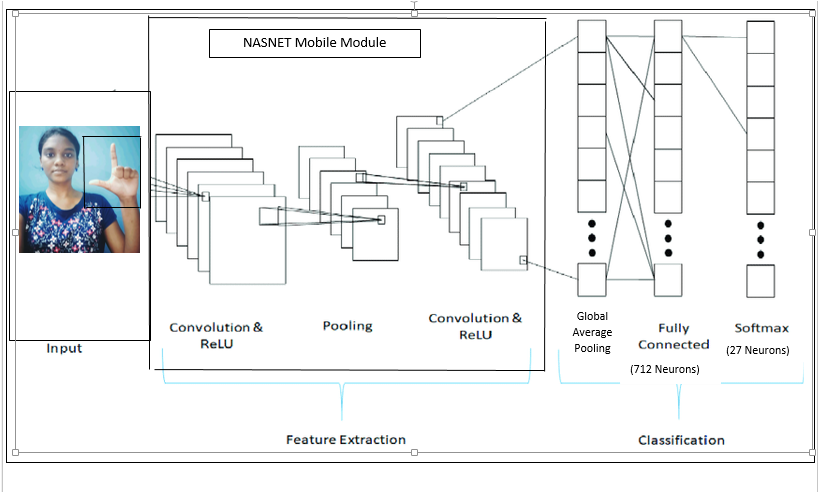
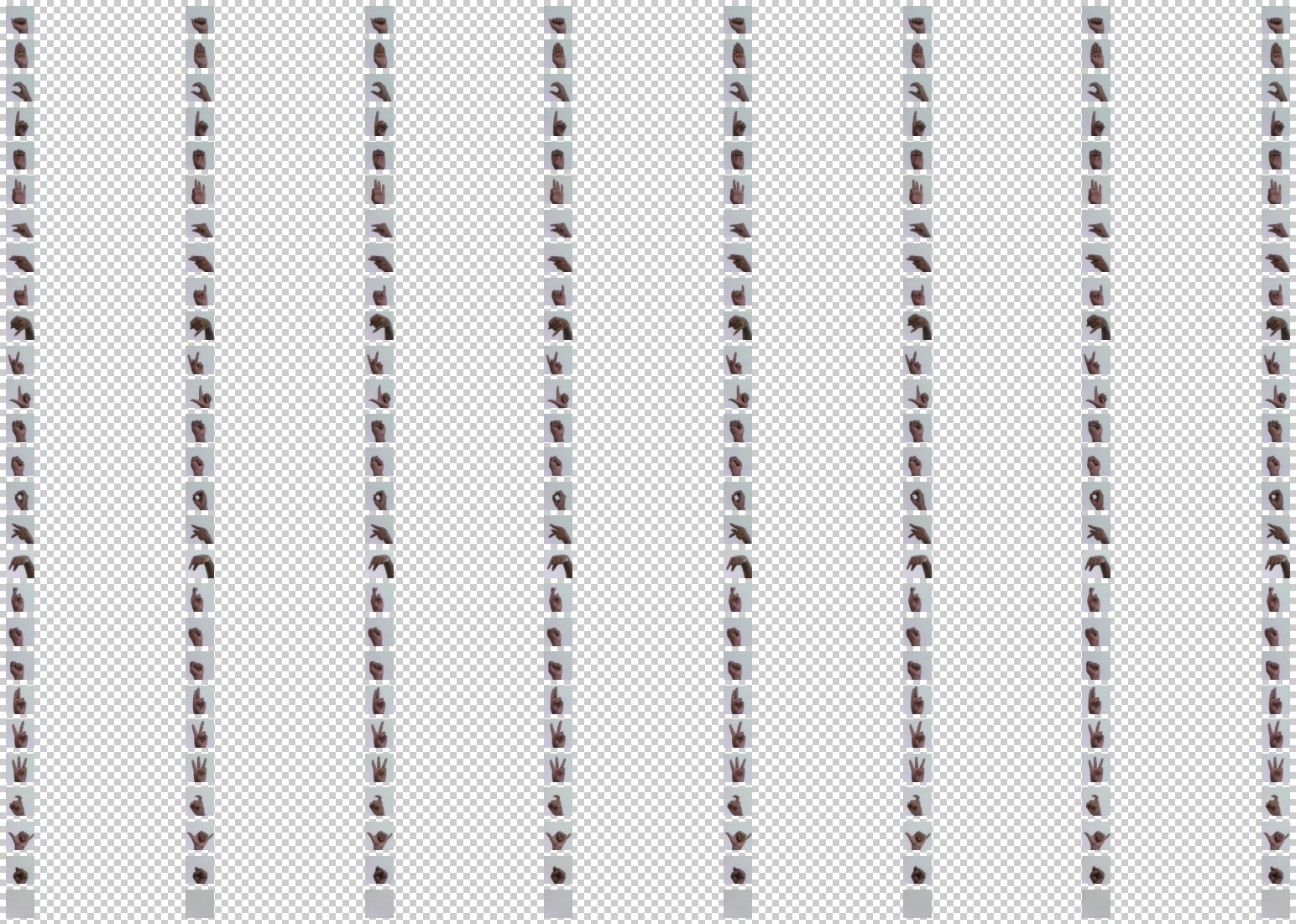


Fig-2 (*Architecture*)

The data is pre-processed by combining the labels of all classes together and converting the integer labels into one hot format. We normalize the images by dividing them with 255.Several python image transformation functions such as Zoom range, Horizontal Flip, Rotation\_range, etc. were applied in order to customize the dataset to fit the necessary input specification (224x224) for the images fed to the pre-trained NASNetMobile model. The input images are expected to have color values in the range [0,1].The data loader was prepared to obtain training and test datasets. Here 85% of the data is used for training and the remaining 20% is used for testing. The training dataset has a total of 2,430 in this 80% of data is used for training and remaining are used for validation. The test dataset has a total of 810 images.



*Fig 3-(Dataset)*

## Classification Method

The feature extraction task in this model was carried out using the convolutional layers of a pre-trained NASNetMobile network. The NASNet is a family of CNN for image classification. It’s architecture comprises alternating sequential convolutional layers, batch normalization layers and pooling layers. The NASNetMobile module contains a trained instance of the network, packaged to transform images into feature vectors and do the image classification. NASNetMobile can achieve very good accuracies while having very low computational costs. The convolutional layer generates output feature maps which are converted to a single dimensional vector by Global Average Pooling(GAP).GAP applies average pooling on the spatial dimensions until each spatial dimension is one, and leaves other dimensions unchanged. In this case values are not kept as they are averaged. It is more effective as it reduces parameters and controls over fitting of the model, by making the model invariant to modulations in images brought forth by different lighting/positions. This improves the generalization of the model. The output we receive here is not in our desired format. So, we add a hidden dense layer with 712 neurons which we train. The activation function applied to introduce non-linearity in the feature maps outputted by the convolutional layer is Rectified Linear Unit (RELU). To reduce over fitting we dropout 40% of the activations. The final output layer which we trained contains 27 output units (since we have 27 classes) with softmax function. Using these a full model is constructed. The final model will have 773 layers.

The code was executed on a Jupyter Notebook. The steps involved in the implementation process were namely,

(1) Creating a custom dataset for letter classes

(2) Visualize the dataset

(3) Pre-processing the data and prepare training and testing sets

(4) Preparing the model for fine tuning

(5) Training the model

(6) Checking Accuracy, Loss graphs during training

(7) Testing the model and saving it

(8) Loading Pre-trained model

(9) Testing on Live Webcam Feed

The hyper parameters used for training the model are as mentioned in the table below.

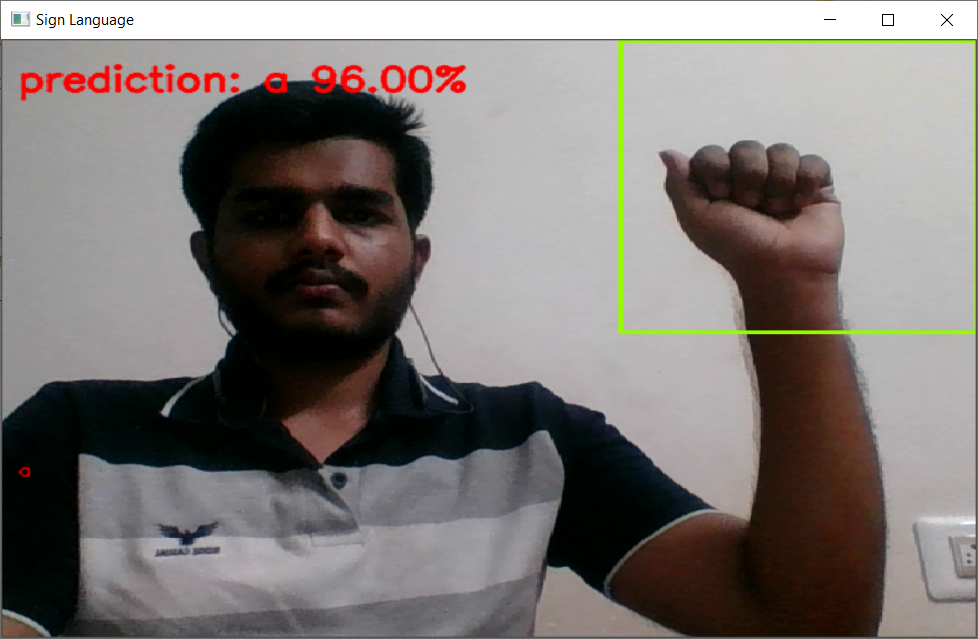
|  |  |
| --- | --- |
| Batch size (training) | 20 |
| Number of epochs | 20 |
| Steps per epoch | 103 |
| Loss Function | Categorical Cross Entropy Loss |
| Optimizer | Adam |
| Learning Rate | 0.0001 |

*Hyper Parameters*

The hyper-parameters were decided upon by continuous experimentation. The batch size were set according to our system. Categorical Cross Entropy was decided as the loss function since the model tries to solve a multi-class classification problem. Adam is an efficient optimizer that proves to be useful in handling sparse gradients, and hence was chosen. The learning rate of Adam is 0.0001, and this default rate is employed in training the model.

# Word Recognition

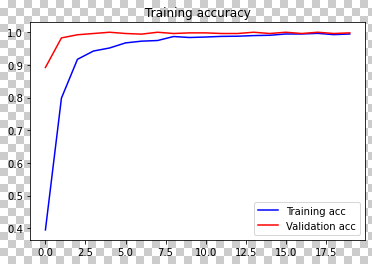
Finally we test our model by running the program. A window pops up with live web-cam feed, here we show a word in sign language. Each letter we show is predicted on the screen along with the probability of the prediction. The letters are displayed one by one. Space can be indicated by an empty image. On finishing we can press ‘a’, the word we showed through hand gestures is voiced out using python text to speech (pyttsx3) feature and it gets saved. We can do this process any number of times. To quit the window press ‘q’.



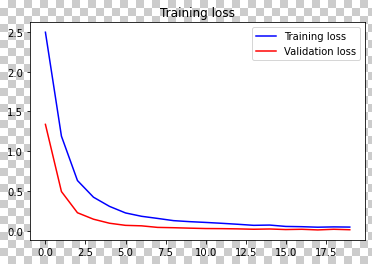
  *Fig4.Word Recognition*

# Results

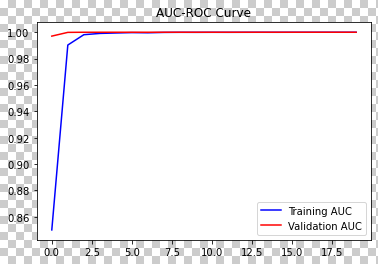
All The graphs are plotted using both training and validation data. The accuracy curve is plotted for the model is shown in Fig.5. As seen, the curves increases exponentially over time which is desirable. The Train Learning Curve calculated from the training dataset gives an idea of how well the model is learning. The Validation Learning Curve calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing. The model loss curve plotted is shown in Fig.6 . Here, the curves decreases exponentially over time which is ideal while training the model. The AUC - ROC curve plotted is shown in Fig7 .It is a performance measurement for the classification problems at various threshold settings. As seen, the curve is very high and nearer to 1, so the model is great at distinguishing the classes. It also indicates a higher number of True positives than False negatives. The classification report is represented as given in Fig.8. As seen, the model gives us optimum precision, recall and f1-score values. The accuracy obtained after training is 1.0 which is a very high value. The confusion matrix can be plotted as shown in Fig.9.



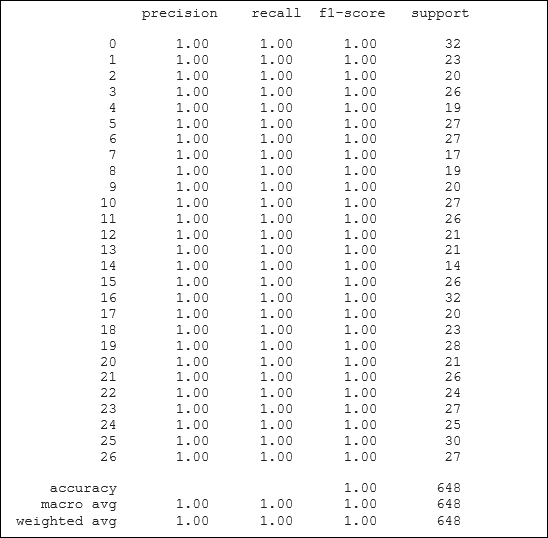
*Fig. 5 Model Accuracy Curve*



*Fig.6 Model Loss Curve*



*Fig.7 AUC-ROC Curve*



*Fig. 8 Classification report*

[[32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27]]

*Fig. 9 Confusion matrix*

# Conclusion

In this project, we have implemented an automatic sign language recognition system in real-time, using tools learnt in computer vision and machine learning. This system gives the chance to deaf people to express their idea without human translator. It also helps them to have an interaction with computer without any help. This paper details the approach that has been employed in achieving the classification task, and also rationalizes the different implementation choices made such as selection of hyper parameters and model architecture. Possible extensions to this project would be ex-tending the gesture recognition system to all other non-alphabet gestures and even continuous sentences in Sign Language.