LOVELY
PROFESSIONAL
UNIVERSITY

SIGN LANGUAGE RECOGNITION

SUBMITTED BY

NAME : PRAVEEN KUMAR S
REG NO : 11809946
SUBJECT : INT 248

SUBMITTED TO

FACULTY NAME : MD. IMRAN HUSSAIN
FACULTY ID : 26819

Contents

Abstract

Sign language is one of the oldest and most natural form of language for communication, but since most people do not know sign language and interpreters are very difficult to come by I have come up with a real time method using neural networks for fingerspelling based american sign language. In our method, the hand is first passed through a filter and after the filter is applied the hand is passed through a classifier which predicts the class of the hand gestures. Our method provides 95.6 % accuracy for the 26 letters of the alphabet.

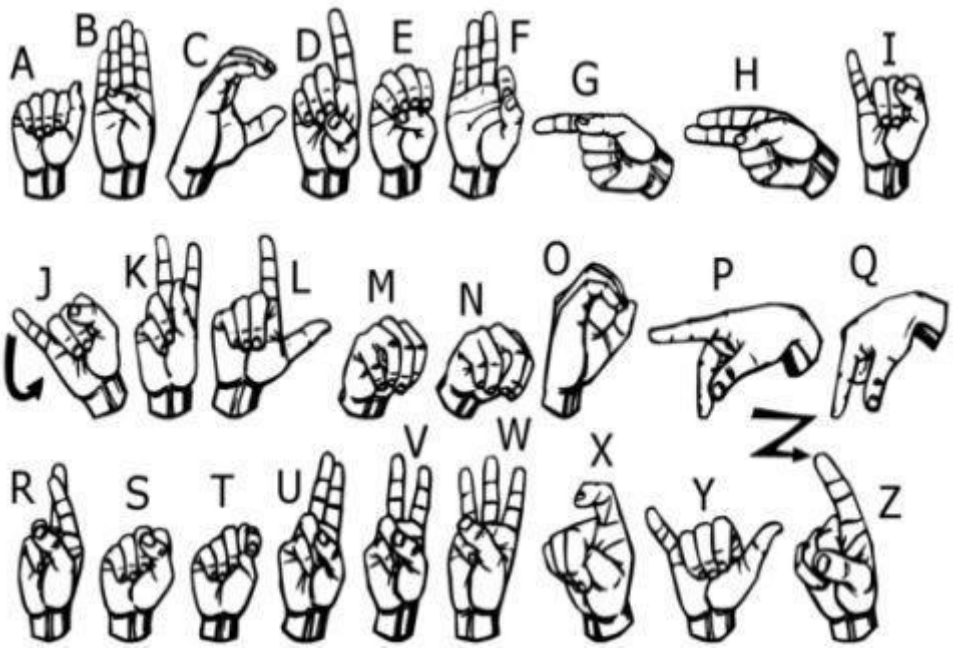
Introduction

American sign language is a predominant sign language Since the only disability D&M people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. Deaf and dumb(D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

Sign language is a visual language and consists of 3 major components[6]:

Fingerspelling	Word level sign vocabulary	Non-manual features
Used to spell words letter by letter .	Used for the majority of communication.	Facial expressions and tongue, mouth and body position.

In our project I basically focus on producing a model which can recognise Fingerspelling based hand gestures in order to form a complete word by combining each gesture. The gestures I aim to train are as given in the image below.

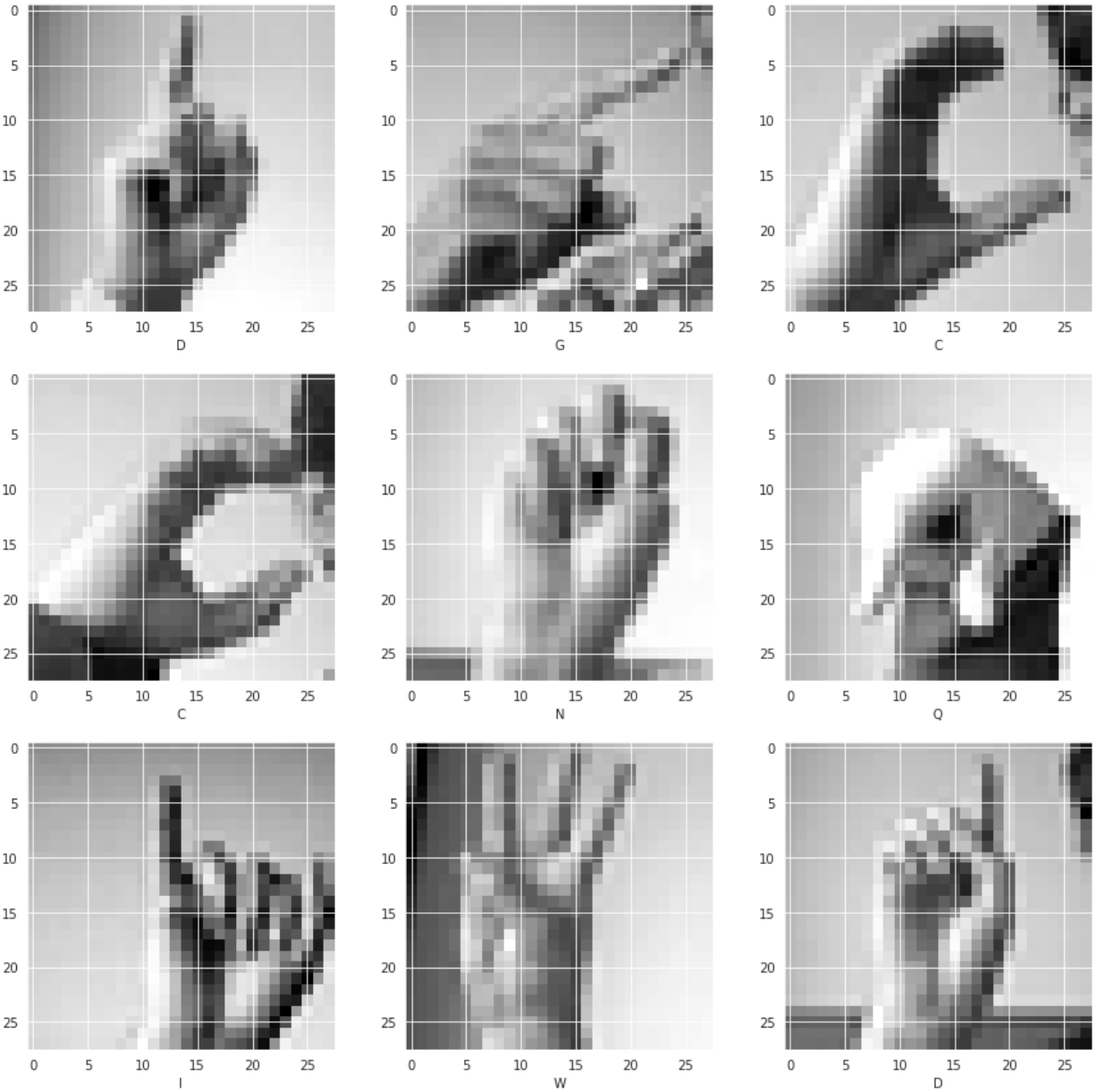


The Data Set

In this article, I have used the American Sign Language (ASL) data set that is provided by MNIST and it is publicly available at Kaggle. This dataset contains 27455 training images and 7172 test images all with a shape of 28 x 28 pixels. These images belong to the 25 classes of English alphabet starting from A to Y (No class labels for Z because of gesture motions). The dataset on Kaggle is available in the CSV format where training data has 27455 rows and 785 columns. The first column of the dataset represents the class label of the image and the remaining 784 columns represent the 28 x 28 pixels. The same paradigm is followed by the test data set.

VISUALIZATION AND PREPROCESSING:

Further preprocessing and visualization, i converted the data frames into arrays.



Keywords and Definition

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. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN

architecture i will reduce the full image into a single vector of class scores

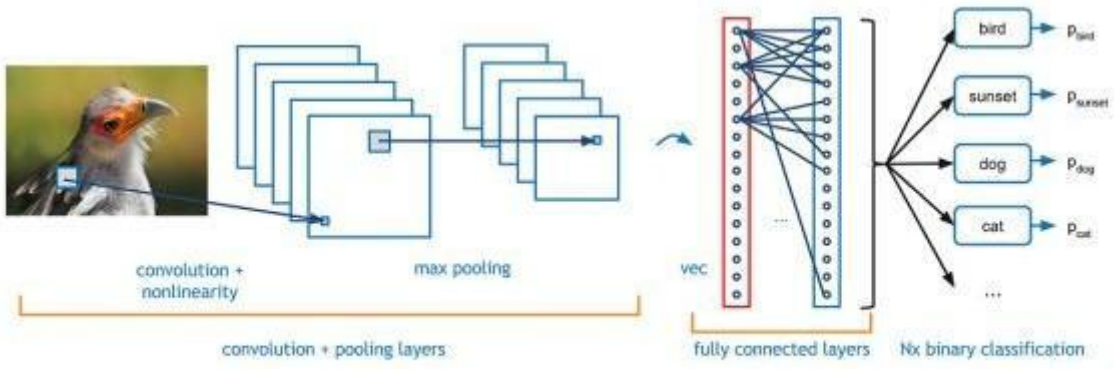


Figure 5.2: Convolution neural networks

1. Convolution Layer : In convolution layer i take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration i slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As i 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

1. Pooling Layer : i 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 i 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 a activation matrix half of its original Size.

b) Average Pooling : In average pooling I take average of all

Values in a window.

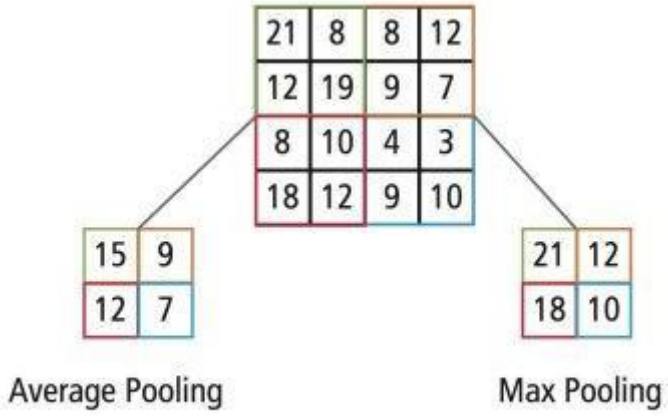


Figure 5.3: Types of pooling

2. 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.

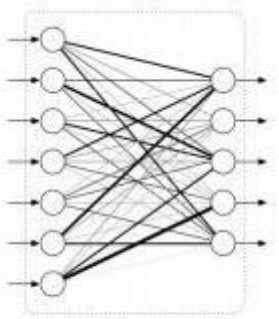


Figure 5.4: Fully Connected Layer

3. 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.

TensorFlow :

Tensorflow is an open source software library for numerical computation. First I define the nodes of the computation graph, then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

Keras :

Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

PROPOSED ARCHITECTURE :

CNN

1. **1st Convolution Layer :**The input picture has resolution of 28x28 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 26X26 pixel image, one for each Filter-weights.
2. **1st Pooling Layer :** The pictures are downsampled using max pooling of 2x2 i.e I keep the highest value in the 2x2 square of array. Therefore, our picture is downsampled to 13x13 pixels.
3. **2nd Convolution Layer :**

   Now, these 13 x 13 from the output of the first pooling layer is served as an input to the second convolutional layer.It is processed in the second convolutional layer using 64 filter weights (2x2pixels each).This will result in a 11 x 11 pixel image.
4. **2nd Pooling Layer :**

   The resulting images are downsampled again using max pool of 2x2 and is reduced to 5 x 5 resolution of images.
5. **3rd Convolution Layer :**

   Now, these 5 x 5 from the output of the first pooling layer is served as an

input to the second convolutional layer.It is processed in the second convolutional layer using 64 filter weights (2x2pixels each).This will result in a 3 x 3 pixel image.

5. **3rd Pooling Layer :**

The resulting images are downsampled again using max pool of 2x2 and is reduced to 1 x 1 resolution of images.

6. **1st Densely Connected Layer :**

Now these images are used as an input to a fully connected layer with 512 neurons and the output from the third convolutional layer is reshaped to an array of 68800 values. The input to this layer is an array of 66048 values. The output of these layer is fed to the 2nd Densely Connected Layer.i am using a dropout layer of value 0.25 to avoid overfitting.

7. **2nd Densely Connected Layer :**

Now the output from the 1st Densely Connected Layer are used as an input to a fully connected layer with 25 neurons.

8. **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 are classifying (alphabets + blank symbol).

Activation Function :

i have used ReLu (Rectified Linear Unit) in each of the layers(convolutional as well as fully connected neurons). ReLu calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features.It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

Pooling Layer :

i apply **Max** pooling to the input image with a pool size of (2, 2) with relu activation function.This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

Dropout Layers:

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples.This layer "drops out" a random set of activations in that layer by setting them to zero.The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out.

Optimizer :

I have used Adam optimizer for updating the model in response to the output of the loss function. Adam combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm(ADA GRAD) and root mean square propagation(RMSProp)

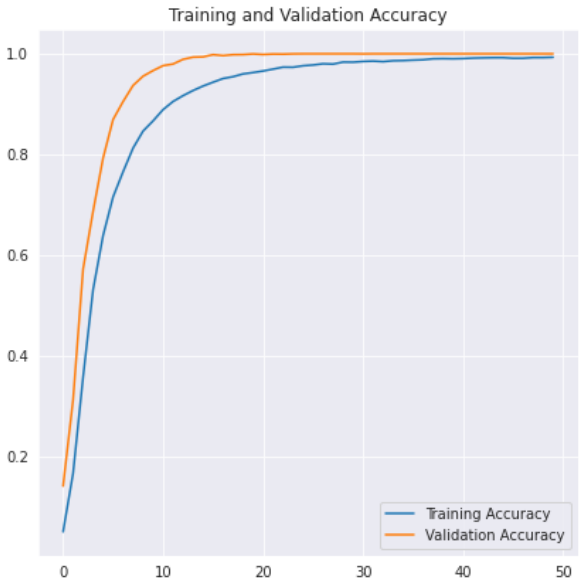
I feed the input images after preprocessing to our model for training and testing after applying all the operations mentioned above.

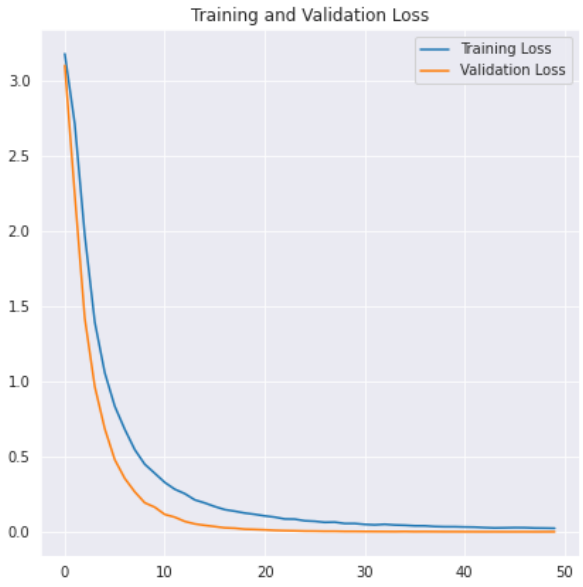
TRAINING AND TESTING:

The prediction layer estimates how likely the image will fall under one of the classes. So the output ARCHITECTURE is normalized between 0 and 1 and such that the sum of each values in each class sums to 1. i have achieved this using softmax function.

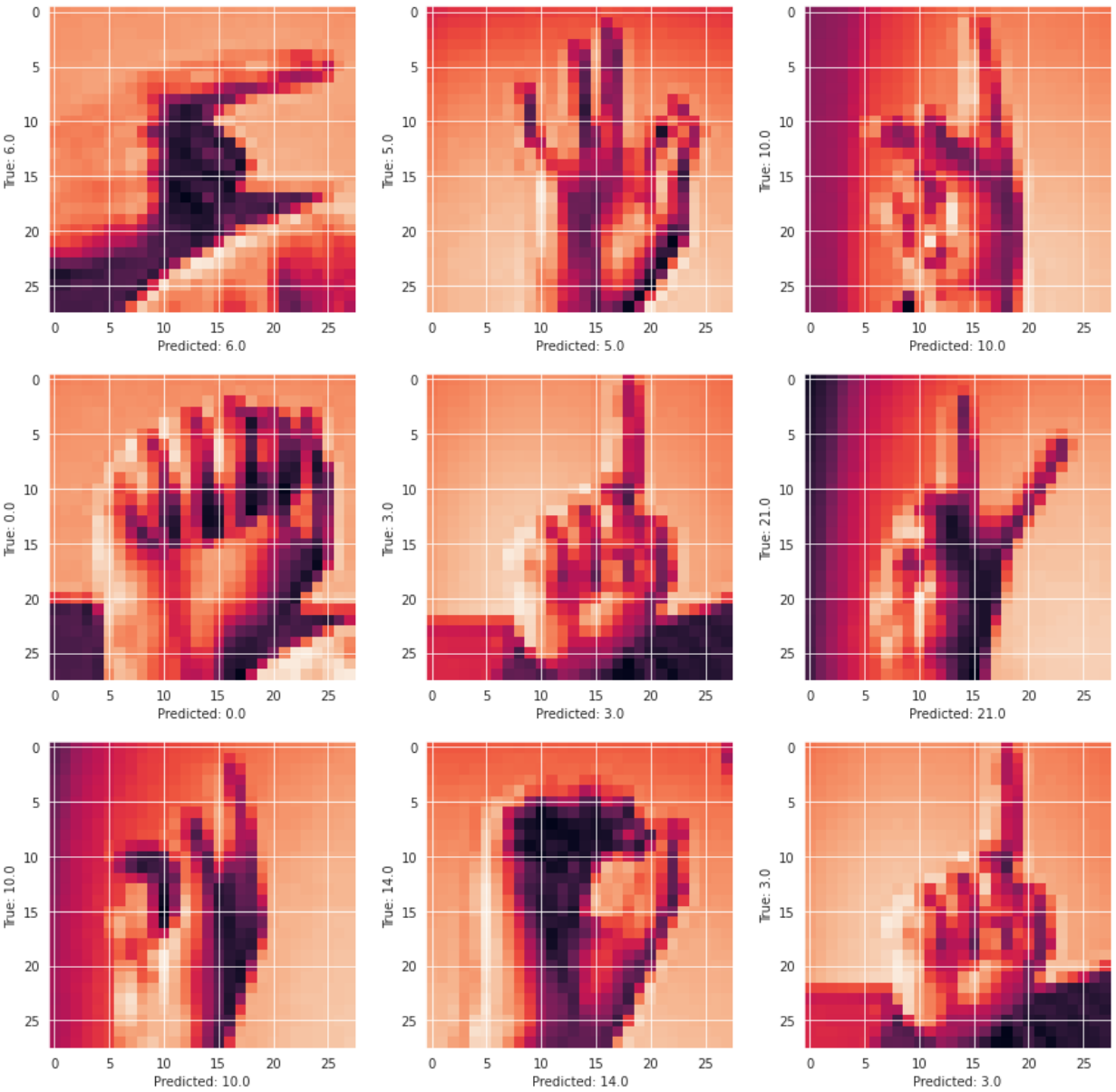
At first the output of the prediction layer will be somewhat far from the actual value. To make it better i have trained the networks using labeled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labeled value and is zero exactly when it is equal to the labeled value. Therefore i optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer i adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy.

As i have found out the cross entropy function, i have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called Adam Optimizer.





PREDICTION :

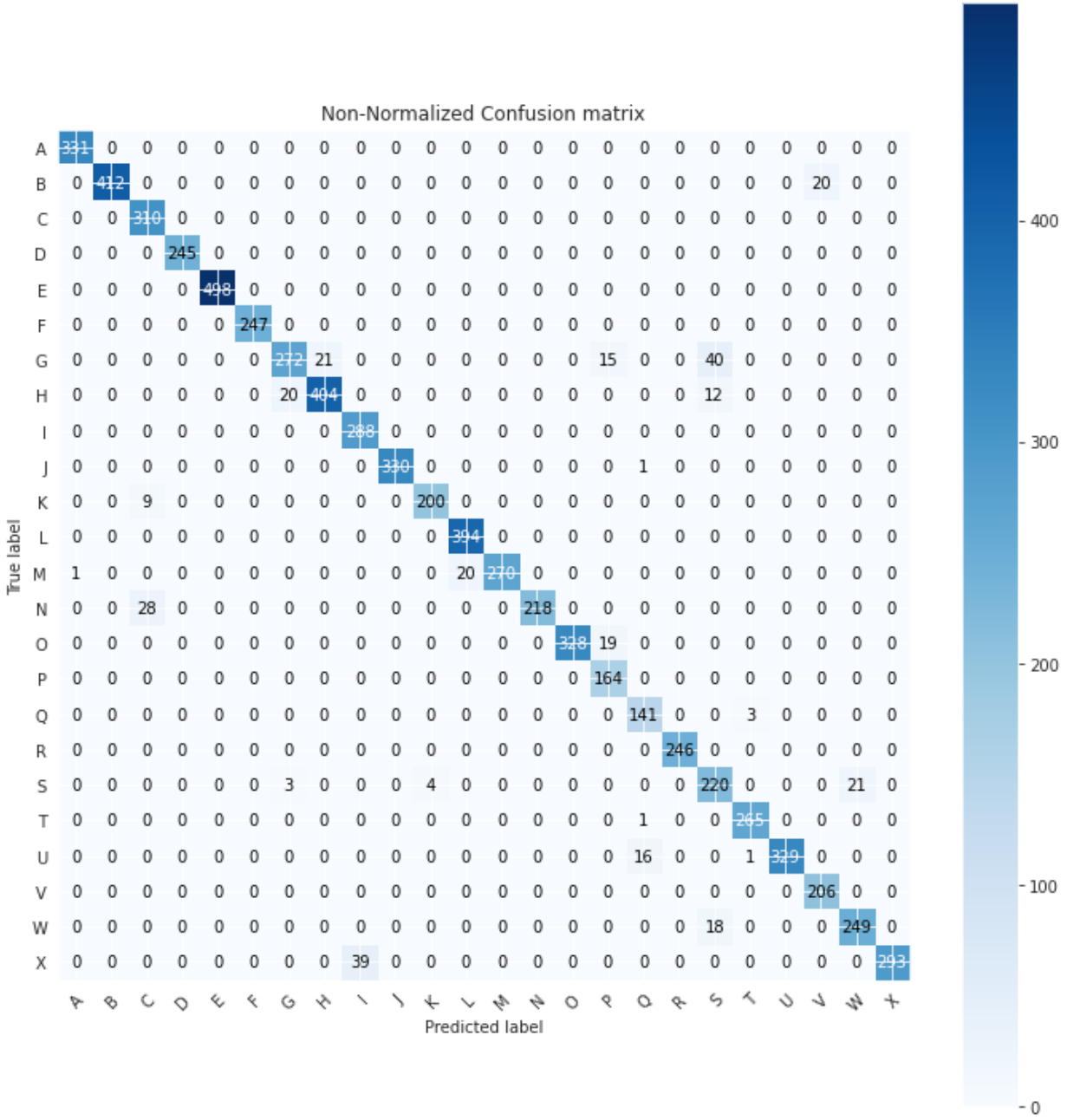

True: 6.0
Predicted: 6.0
True: 5.0
Predicted: 5.0
True: 10.0
Predicted: 10.0
True: 0.0
Predicted: 0.0
True: 3.0
Predicted: 3.0
True: 21.0
Predicted: 21.0
True: 10.0
Predicted: 10.0
True: 14.0
Predicted: 14.0
True: 3.0
Predicted: 3.0

Challenges Faced :

There were many challenges faced by us during the project. The very first issue i faced was of dataset. i wanted to deal with raw images and that too square images as CNN in Keras as it was a lot more convenient working with only square images. i found existing dataset for that hence i decided to make our project. Second issue was to select a filter which i could apply on our images so that proper features of the images could be obtained and hence then i could provided that image as input for CNN model. More issues were faced relating to the accuracy of the model i trained in earlier phases which i eventually improved by increasing the input image size and also by improving the dataset.

Results :

i have achieved an accuracy of 95.6% in our model using only cnn of our algorithm , which is a better accuracy then most of the current research papers on american sign language.



Conclusion :

In this report, a functional real time vision based american sign language recognition for D&M people have been developed for asl alphabets.i achieved final accuracy of 95.6% on our dataset. i are able to improve our prediction after implementing layers of algorithms in which i verify and predict symbols which are more similar to each other.

This way i am able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

Future Scope :

i are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. i also thinking of improving the preprocessing to predict gestures in low light conditions with a higher accuracy.

References :

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

[2] https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_b ilateral_filter/gausian_median_blur_bilateral_filter.html

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] Guide To Sign Language Classification Using CNN

[5] aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/

[6] https://en.wikipedia.org/wiki/Convolutional_neural_network

PROJECT LINK :

https://github.com/praveen-kumars/Sign-Language-Recognition