GESTURE CLASSIFICATION

The approach which we used for this project is : Our approach uses two layers of algorithm to predict the final symbol of the user.

**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.

**Layer 1:**

CNN Model :

1. 1st Convolution Layer :The input picture has resolution of 128x128 pixels.

It is first processed in the first convolutional layer using 32 filter weights

(3x3 pixels each). This will result in a 126X126 pixel image, one for each

Filter-weights.

2. 1st Pooling Layer : The pictures are downsampled using max pooling of

2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our

picture is downsampled to 63x63 pixels.

3. 2nd Convolution Layer :

Now, these 63 x 63 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 32 filter weights (3x3 pixels each).This will result

in a 60 x 60 pixel image.

4. 2nd Pooling Layer :

The resulting images are downsampled again using max pool of 2x2 and is

reduced to 30 x 30 resolution of images.

5. 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 convolutional layer is reshaped to an

array of 30x30x32 =28800 values. The input to this layer is an array of

28800 values. The output of these layer is fed to the 2nd Densely Connected

Layer.We are using a dropout layer of value 0.5 to avoid overfitting.

6. 2nd Densely Connected Layer :

Now the output from the 1st Densely Connected Layer are used as an input

to a fully connected layer with 96 neurons.

7. 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 + blaActivation Function :

We 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 :**

We 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[5].

**Optimizer :**

We 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)nk symbol).

**Layer 2:**

**We are using two layers of algorithms to verify and predict symbols which**

**are more similar to each other so that we can get us close as we can get to**

**detect the symbol shown. In our testing we found that following symbols**

**were not showing properly and were giving other symbols also :**

**1. For D : R and U**

**2. For U : D and R**

**3. For I : T, D, K and I**

**4. For S : M and N**

**So to handle above cases we made three different classifiers for classifying**

**these sets:**

**1. {D,R,U}**

**2. {T,K,D,I}**

**3. {S,M,N}**