**readme**

**B19CSE039\_B19CSE042\_B19CSE045**

**This project is to show the importance and Comparison of SGD-SVM , CNN and MLP**

**MLP:**

**Preprocessing:**

1)The feature set was converted into categorical integers representing each class in the dataset.

2)The training and testing feature sets were reshaped from numpy arrays containing pixel info as 32\*32\*3 where 32\*32 were pixel values of the 32\*32 image and 3 were the RGB values, into a single 50000,3072 shaped array with all the pixel information and the RGB info is contained in one row of the array note 3arrays within array. This was done to suit the format in which the keras(tensorflow library) designed sequential model accepts input.

3) The input was normalised by dividing every pixel value by 255 in the feature set

**Model Building:**

1)Two ‘dense’ forms of layers were added to the network. Dense layers are the most common form of layers which perform the following function on the input:

output = activation(dot(input, kernel) + bias)

2)Here for the first two layers, the number of neurons was chosen to be 256 with the activation function chosen as the most common type “Rectified Linear Unit” function.

3)The first layer also contains the dimensions of the input the layer will receive which will be (Batch\_size,3072) where batch size will be given at the time of running.

4)The final layer contains 10 neurons and the activation function is chosen as softmax because “softmax is used as the activation function for multi-class classification problems.

5) Next we’ve defined the SGD optimizer as the optimizer for our network with the standard learning rate of 0.01 and momentum of 0.9. We have chosen nesterov to be true.

6)The model was compiled to use SGD as the optimizer with categorical\_cross\_entropy as the loss function to be minimised and val\_accuracy to be maximized.

7)The model was trained for 15epochs

**CNN:**

**Preprocessing:**

1)The feature set was converted into categorical integers representing each class in the dataset.

2)The training and testing feature sets were reshaped from numpy arrays containing pixel info as 32\*32\*3 where 32\*32 were pixel values of the 32\*32 image and 3 were the RGB values, into a single 50000,3072 shaped array with all the pixel information and the RGB info is contained in one row of the array note 3arrays within array. This was done to suit the format in which the keras (tensorflow library) designed sequential model accepts input.

3) The input was normalised by dividing every pixel value by 255 in the feature set

**Model Building:**

1)A sequential model type was chosen. A 2D convolution layer of the shape 32,3,3 was applied on the input layer with Relu as the chosen activation function

2)In the first layer, the shape of the input was given as it is in the from 32\*32\*3.

3) A second convolutional layer was added with the same Relu activation function and similar shape.

4) A max pooling of the above convolutional layer was done.

5) Next the the output of this second layer was flattened so it can be sent to the next layer for implementing activation function and assigning weights and biases

6) Next layer is defined as 256 neurons with ReLU as their activation function and the output of the layer above as their input

7)The final layer is assigned with 10 neurons and softmax as the activation function for reasons similar to MLP.

8)SGD was chosen as the optimizer with the standard learning rate,decay rate and momentum with nesterov set to True.

9)The model was compiled to minimize the categorical\_loss\_entropy loss function while trying to maximize the val\_accuracy.

10)The model was ran for 10epochs with the validation split of 0.2 to determine the validation accuracy

**Improving:**

1)In an effort to improve the accuracy of the model two drop out layers were added after the max pooling layer and the first neuron assignment layer.

2)The model was again run with the similar optimizer,epochs,loss function and batch size etc.

3)In an effort to further optimize the model batch normalization was done after the second convolutional layer and before the second dropout layer.

4)The model was again run with the similar optimizer,epochs,loss function and batch size etc.

**SGD-SVM:**

We have shown confusion matrices, accuracies, precision,losses,recall and f1 score throughout the project.

The most important and necessary codes are explained for SGD-SVM part:

Skimage.color.rgb2gray lets us convert the image to grayscale image since it needs to convert to a fog image in the upcoming code.

We need to import hog from skimage.feature and we can then use them to create a hog image that produces a black and white outline which is often observed to be an important preprocessing step to improve the accuracy.

There are several hyperparameters such as

* pixels\_per\_cell
* cells\_per\_block
* Orientations

that are to be manually decided for the code for better improvement.

THerefore 10cases f different combinations of these hyperparameters were taken into account for improving the model with the best set of parameters.

Once the hog classification is over, we then need to Standardise and that is done using standardscaler().

We assign it to scalify and fit\_transform the training data as a form of conversion.

SGDClassifier uses SVM by-default.Therefore , we need not mention it inside the code.

Then,we fit the data and predict it.

SGDClassifier was used to fit and predict the data.

At last,

from mlxtend.plotting import plot\_confusion\_matrix

A confusion matrix was plotted to identify if the majority of the classes were correctly classified or not using the above library.

Precision ,recall , accuracy\_score ,f1\_score were all calculated with the help of in-built sklearn libraries.