Data Analytics and Visualization

DS250

Assignment 1b

Animesh Raj : 11940120

Puja Bansal : 11940910

Flattening: An abstraction of an image used to characterize and numerically quantify the contents of an image. Characterizes as real, integer, or binary valued. Simply put, a feature vector is a list of numbers used to represent an image. Converting the 2D feature matrix into a 1D array feature vector.

Color Histogram: Create RGB histogram from an image by setting the number of bins by default to (8,8,8) into which the red, green and blue channels are to be divided. (RGB default value set to 256).

HOG: It is a technique that counts events of gradient orientation in a specific portion of an image or region of interest. It extracts the information of the edges magnitude as well as the orientation of the edges.

SIFT: It is a feature extraction method where image content is transformed into local feature coordinates that are invariant to translation, scale and other image transformation.

Part - a

Learning with prototype (LWP) →

https://colab.research.google.com/drive/1Mn-ih87_oEsP7FJOT7Dccd7jH3weurkU

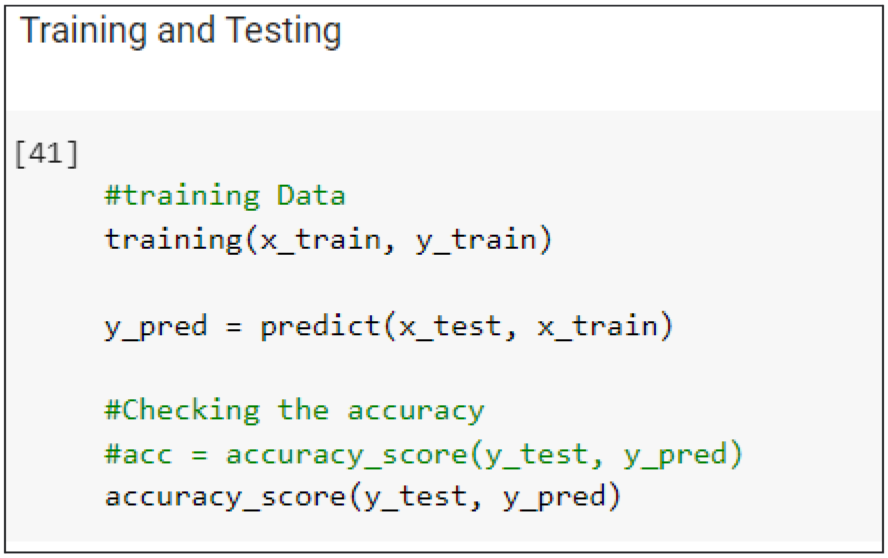
Def: It represents each class by a “prototype” vector and takes out the “mean” or “average” of inputs from that class and predicts the label of each test input based on its distances from the class prototypes and then the predicted label will be the class that is the closest to the test input.

Feature extraction technique: Flattening, Histogram, HOG

Training images: 40000 (Taken approx 300 images per directory)

Test images: 5000

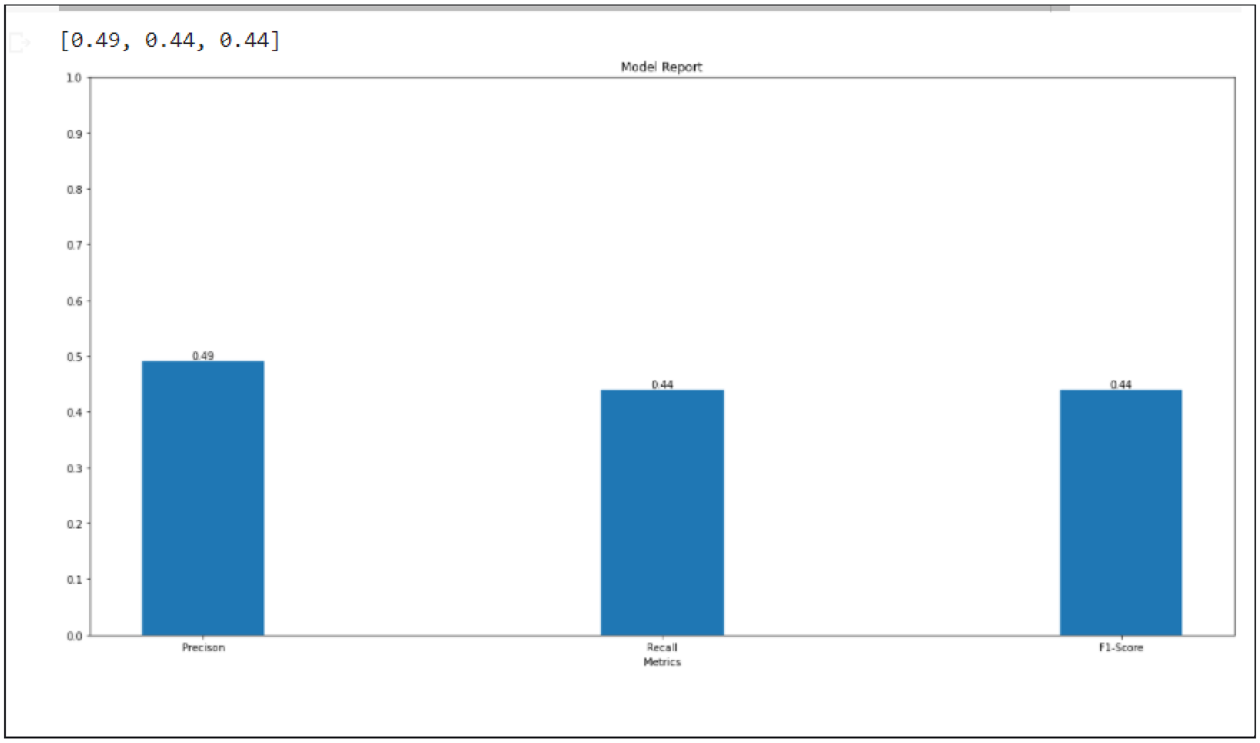
Accuracy: 0.44214543993571714 = 44.21%


Training and Testing

```
[41]
    #training Data
    training(x_train, y_train)

    y_pred = predict(x_test, x_train)

    #Checking the accuracy
    #acc = accuracy_score(y_test, y_pred)
    accuracy_score(y_test, y_pred)
```



```
[0.49, 0.44, 0.44]
```

Model Report

Part(b) :

K-Nearest Neighbor (KNN) →

https://colab.research.google.com/drive/1wUSnI9UgXsffZ5tjg1T1c_fxXzKMdvk8

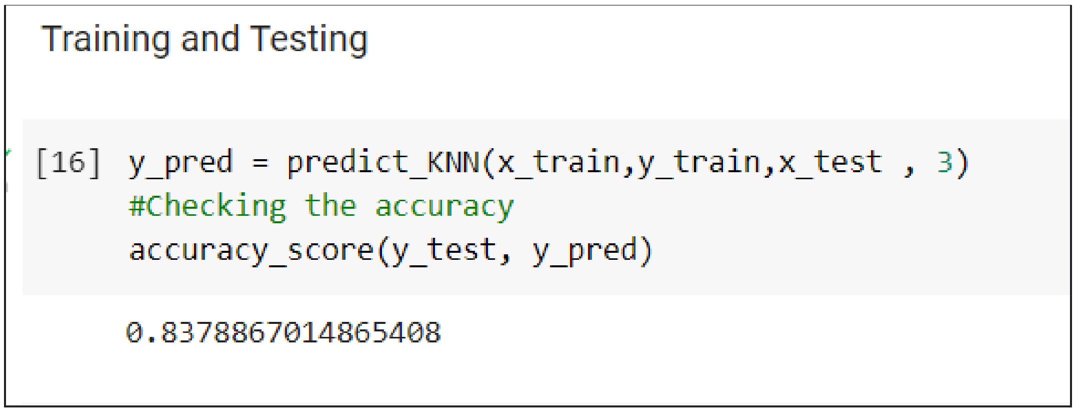
Def: K-Nearest Neighbor is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classify a data point based on how its neighbors are classified.

Feature extraction technique: Flattening, Histogram, HOG

Training images: 40000 (Taken approx 300 images per directory)

<u>Test images:</u> 5000

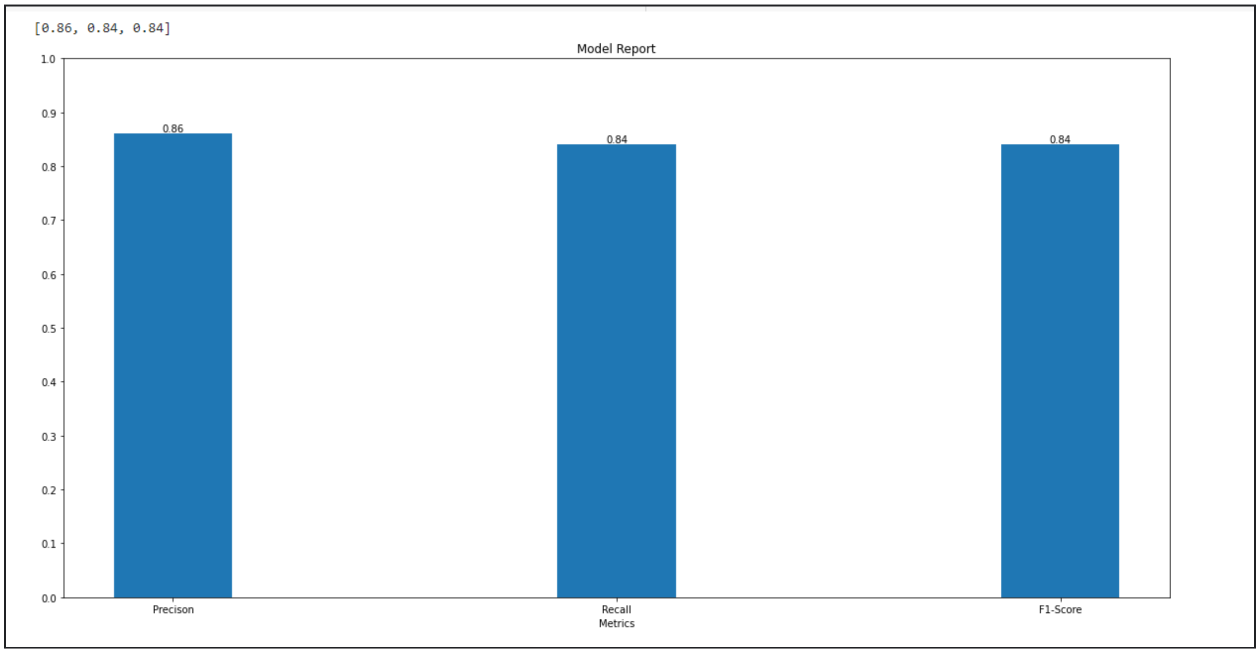
<u>Accuracy:</u> 0.8378867014865408 = 83.78%



Training and Testing

```
[16] y_pred = predict_KNN(x_train,y_train,x_test , 3)
     #Checking the accuracy
     accuracy_score(y_test, y_pred)

     0.8378867014865408
```



```
[0.86, 0.84, 0.84]
```

Model Report

HOW CAN WE BREAK TIES?

If there is a tie, we can use further techniques to filter the result like simply using Learning with Prototype amongst the tied data points to find out the result.

Part(c)

Multi-Layer perceptron →

https://colab.research.google.com/drive/1RGLhjzlc3L3tHmnZj9VyCjf32TstFPmH

Def: A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Optimizer: This could be a string identifier of an existing optimizer (such as adam), or an instance of the optimizer class.

Loss Function: This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as categorial_crossentropy), or it can be an objective function.

Metrics: A metric could be the string identifier of an existing metric (such as accuracy), or a custom metric function.

Epochs: One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE. Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

→ We are taking epochs value as 200, so that for the entire data set, it back propagates several(200) times to decrease the loss and to get better accuracy.

Relu: This activation function is used in hidden layers and is effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. It does not activate all the neurons at the same time.

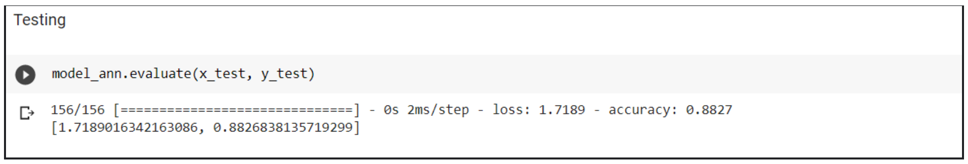
SoftMax: The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. That is, softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels. We didn't use it as a function for hidden layers because it will keep all the nodes (hidden variables) linearly dependent which may result in many problems and poor generalization.

Feature extraction technique: Flattening, Histogram, HOG

Training images: 40000 (Taken approx 300 images per directory)

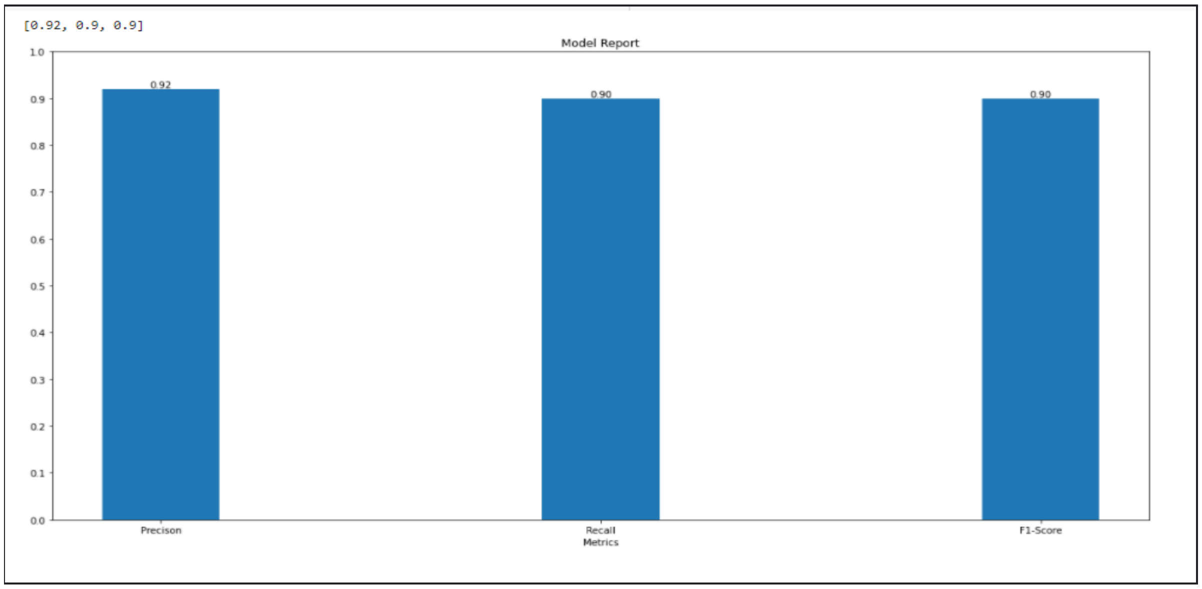
Test images: 5000

Accuracy: 0.8826838135719299 = 88.26%


Testing

```
model_ann.evaluate(x_test, y_test)
```

```
156/156 [==============================] - 0s 2ms/step - loss: 1.7189 - accuracy: 0.8827
[1.7189016342163086, 0.8826838135719299]
```



```
[0.92, 0.9, 0.9]
```

Problem Faced : →

- ❖ For better results, we tried to use multiple feature extraction techniques but unfortunately failed while using SIFT. (patented error of OpenCV, cv2).
- ❖ When we were adding layers to ANN, there was a decrease in accuracy.
- ❖ We were just able to train max for 40000 images and not for the whole data set. As it was taking a lot of time and RAM too got crashed several times.