Link for github: https://github.com/sahersohail013/computervision

Task 1:

We were supposed to compute the HOG features and train classifiers on it i.e., SVM and Random Forest. HOG is an algorithm which helps with object detection in computer vision and image processing models. HOG is a kind of feature descriptor that counts occurrences of gradient orientation in localized portions of an image. It was observed that is the parameters of the models were tweaked the F1 score, precision and recall of models changed respectively.

```
#TEST THE MODELS

# Create predictions(SVM)
pred_labels = SVCmod.predict(testdata)
print("Prediction is done")
#CLASSIFICATION FOR MODELS
from sklearn.metrics import classification_report
print(classification_report(testlabel, pred_labels))
```

```
Prediction is done
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1132
           1       1.00      1.00      1.00       453

    accuracy                           1.00      1585
   macro avg       1.00      1.00      1.00      1585
weighted avg       1.00      1.00      1.00      1585
```

```
#TEST THE MODELS

# Create predictions(SVM)
pred_labels = SVCmod.predict(testdata)
print("Prediction is done")
#CLASSIFICATION FOR MODELS
from sklearn.metrics import classification_report
print(classification_report(testlabel, pred_labels))
```

```
Prediction is done
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1132
           1       1.00      1.00      1.00       453

    accuracy                           1.00      1585
   macro avg       1.00      1.00      1.00      1585
weighted avg       1.00      1.00      1.00      1585
```

```
# Create predictions(RANDOM FOREST)
pred_labelsRF = randomforest.predict(testdata)
print("PredictionRF is done")
print(classification_report(testlabel, pred_labelsRF))
```

```
PredictionRF is done
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1132
           1       1.00      1.00      1.00       453

    accuracy                           1.00      1585
   macro avg       1.00      1.00      1.00      1585
weighted avg       1.00      1.00      1.00      1585
```

```
# Create predictions(RANDOM FOREST)
pred_labelsRF = randomforest.predict(testdata)
print("PredictionRF is done")
print(classification_report(testlabel, pred_labelsRF))
```

```
PredictionRF is done
              precision    recall  f1-score   support

           0       0.71      1.00      0.83      1132
           1       0.00      0.00      0.00       453

    accuracy                           0.71      1585
   macro avg       0.36      0.50      0.42      1585
weighted avg       0.51      0.71      0.60      1585
```

While visualizing the images most of the images it was found that there is no difference between the actual and predicted results which means the accuracy of the model is 100% however for some images the model misclassifies

```
i = 103
#get image
image = testdata_[i]

#compute hog feature vector for above image
(hog, hogimage) = HOG_func(image)
obtained_label = SVCmod.predict([hog])
obtained_labelrf = randomforest.predict([hog])

#comparison
print("Actual Label svm =")
print(testlabel[i])
print("Predicted Labelsvm =")
print(obtained_label[0])
#comparison
print("Actual Label rf =")
print(testlabel[i])
print("Predicted rf =")
print(obtained_label[0])

#visualize
figr, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 20), sharex=True, sharey=True)
ax1.axis('off')
ax1.imshow(image, cmap="gray")
ax1.set_title('Input image')
ax2.axis('off')
ax2.imshow(hogimage,cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()
```

```
Actual Label svm =
0
Predicted Labelsvm =
0
Actual Label rf =
0
Predicted rf =
0
```

```
#VISUALIZATION
i = 99
#get image
image = testdata_[i]

#compute hog feature vector for above image
(hog, hogimage) = HOG_func(image)
obtained_label = SVCmod.predict([hog])
obtained_labelrf = randomforest.predict([hog])

#comparison
print("Actual Label =")
print(testlabel[i])
print("Predicted Label =")
print(obtained_label[0])
print("Actual Label rf =")
print(testlabel[i])
print("Predicted rf =")
print(obtained_label[0])
#visualize
figr, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 20), sharex=True, sharey=True)
ax1.axis('off')
ax1.imshow(image, cmap="gray")
ax1.set_title('Input image')
ax2.axis('off')
ax2.imshow(hogimage,cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()
```

```
Actual Label =
0
Predicted Label =
1
Actual Label rf =
0
Predicted rf =
1
```

Task2:

For this task we have used the Bag of visual words technique. To compute Bag of visual words the first step was to determine the features in an image for a given label (feature extraction). Then we created a visual corpus by clustering and performed frequency analysis. Lastly, we classified the images based on the vocabulary generated by the SVM classifier.

In order to execute the code, we need to fulfill the following requirements:

```
! pip uninstall opencv-python -y
! pip uninstall opencv-contrib-python -y
! pip install opencv-python==3.4.11.45
! pip install opencv-contrib-python==3.4.11.45
! pip install tqdm
```

Objects dataset:

It was noted that by changing the number of clusters from 50 to either less than 50 or more than 50, the accuracy of the model decreased significantly. By keeping the number of clusters in k mean = 50, we get

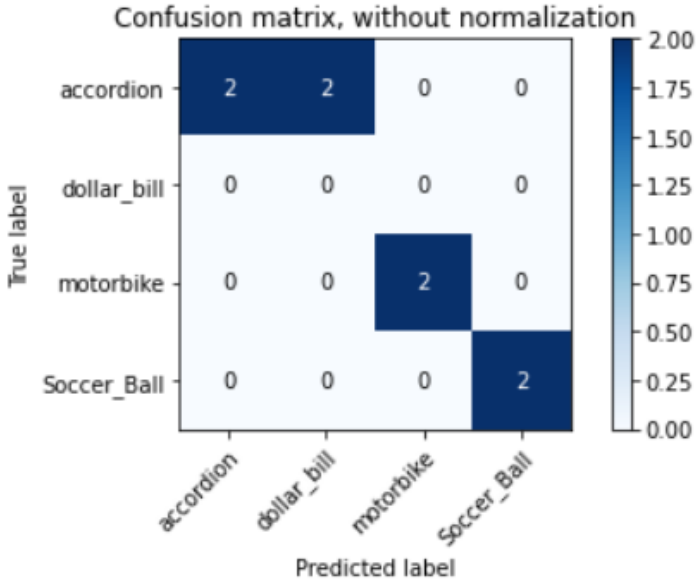
the highest accuracy of 0.750 i.e. 75%. In the future we can increase the accuracy of the model by adding more data, treating missing values, feature engineering, ensemble methods and algorithm tuning.

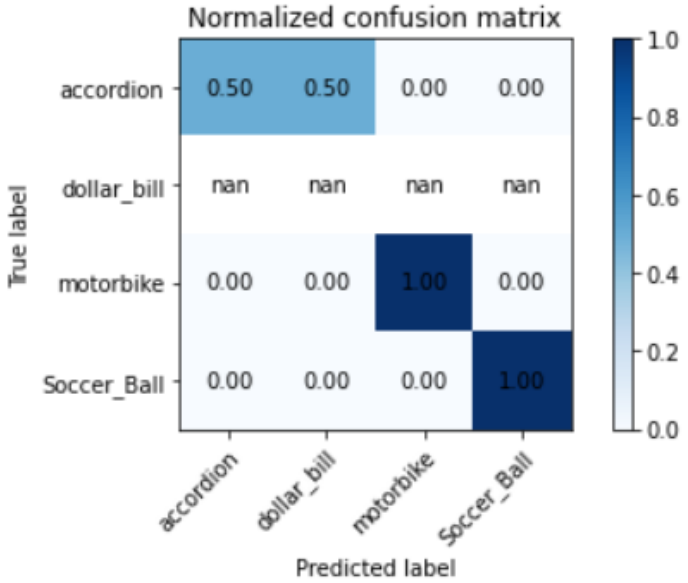
The classification report for this model is given as follows:

	precision	recall	f1-score	support
Soccer_Ball	1.00	0.50	0.67	4
acccordian	0.00	0.00	0.00	0
dollar_bill	1.00	1.00	1.00	2
motorbike	1.00	1.00	1.00	2
accuracy			0.75	8
macro avg	0.75	0.62	0.67	8
weighted avg	1.00	0.75	0.83	8

--- 1.2510433197021484 seconds ---

The confusion matrix after applying SVM is given below:





Flowers dataset:

It was noted that by changing the number of clusters from 50 to either less than 50 or more than 50, the accuracy of the model decreased significantly. By keeping the number of clusters in k mean = 50, we get the highest accuracy of 0.412 i.e 42%. In the future we can increase the accuracy of the model by adding more data, treating missing values, feature engineering, ensemble methods and algorithm tuning.

The classification report for this model is given as follows:

	precision	recall	f1-score	support
daisy	0.53	0.26	0.35	127
dandelion	0.63	0.37	0.46	180
roses	0.50	0.03	0.06	129
sunflowers	0.55	0.49	0.52	140
tulips	0.30	0.82	0.44	160
accuracy			0.41	736
macro avg	0.50	0.39	0.37	736
weighted avg	0.50	0.41	0.38	736

The confusion matrix after applying SVM is given below: