CV assignment-03

-S20170010045 D.Nikhil

-S20170010172 V.Hemanth

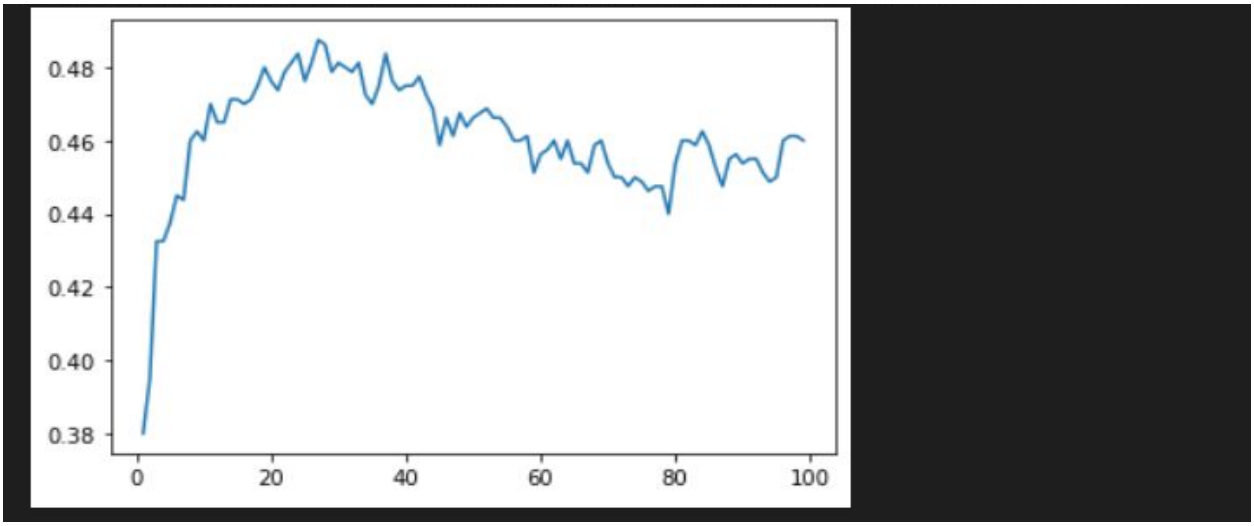
-S20170010102 N.N.Sri Ram

Image categorization

Question-1

In this task we have to implement image features, train a classifier using
the training samples, and then evaluate the classifier on the test set.
We need to implement bag of visual words model and implement Kmeans cluster algorithm to compute visual word dictionary
Steps for implementation:
1.Loading labels of training and test data.
2.extracting features and regions of given data.
3.computing visual bag-of-words dictionary
4.implementing k-means clustering algorithm to features.
5.calculating confusion matrix and accuracy.
Accuracy is 0.47
Confusion matrix is attached below

	precision	recall	f1-score	support
1	0.39	0.53	0.45	100
2	0.69	0.82	0.75	100
3	0.44	0.31	0.36	100
4	0.50	0.45	0.47	100
5	0.37	0.42	0.39	100
6	0.21	0.27	0.24	100
7	0.52	0.43	0.47	100
8	0.44	0.27	0.33	100
accuracy			0.44	800
macro avg	0.45	0.44	0.43	800
weighted avg	0.45	0.44	0.43	800



Scale :

X-Axis : Values of K (Neighbours) from 1 to 100

Y-Axis : Measures of Accuracy

Question-2

1)We are using the pretrained AlexNet. The classifier[6] is changed to 8 to get 8 out_features which was asked to implement.

2) Read the csv files of train and test and create dictionary with image and corresponding labels for it.

3) Initialize the model using initialize_model function where you load the pretrained model of alexnet.

4) We will get the parameters to be optimized or updated. then we will know what parameters to be learned.

5) Training the model and then testing the model and we got 90% accuracy

```
epoch  1  total loss : 381.94107869267464  total correct :  1509  accuracy
: 79.92584745762711
epoch  2  total loss : 323.50325098633766  total correct :  1610  accuracy
: 85.27542372881356
epoch  3  total loss : 247.64012730121613  total correct :  1666  accuracy
: 88.24152542372882
epoch  4  total loss : 265.58933478593826  total correct :  1641  accuracy
: 86.91737288135593
epoch  5  total loss : 284.98790097236633  total correct :  1643  accuracy
: 87.02330508474576
```

```
epoch  6  total loss : 250.1824003458023  total correct :  1669  accuracy
: 88.40042372881356
epoch  7  total loss : 223.3920721411705  total correct :  1693  accuracy
: 89.67161016949152
epoch  8  total loss : 187.0485027730465  total correct :  1725  accuracy
: 91.36652542372882
epoch  9  total loss : 179.31294015049934  total correct :  1716  accuracy
: 90.88983050847457
epoch  10  total loss : 191.97559148073196  total correct :  1712
accuracy : 90.67796610169492
epoch  11  total loss : 135.82300490140915  total correct :  1752
accuracy : 92.79661016949152
epoch  12  total loss : 155.25749826431274  total correct :  1733
accuracy : 91.79025423728814
epoch  13  total loss : 153.37523424625397  total correct :  1741
accuracy : 92.21398305084746
epoch  14  total loss : 165.06358033418655  total correct :  1726
accuracy : 91.41949152542372
epoch  15  total loss : 141.56284427642822  total correct :  1747
accuracy : 92.53177966101694
epoch  16  total loss : 136.85224813222885  total correct :  1747
accuracy : 92.53177966101694
epoch  17  total loss : 129.29159700870514  total correct :  1752
accuracy : 92.79661016949152
epoch  18  total loss : 125.71328336000443  total correct :  1743
accuracy : 92.31991525423729
epoch  19  total loss : 124.69214099645615  total correct :  1764
accuracy : 93.4322033898305
epoch  20  total loss : 132.67240342497826  total correct :  1756
accuracy : 93.00847457627118
```

time taken to train : 231.4899754524231

```
tensor([[95.,  0.,  0.,  0.,  0.,  5.,  0.,  0.],
        [ 0., 88.,  0.,  0.,  1., 11.,  0.,  0.],
        [ 0.,  0., 96.,  0.,  0.,  2.,  2.,  0.],
        [ 0.,  0.,  1., 91.,  0.,  0.,  6.,  2.],
        [ 1.,  0.,  1.,  0., 94.,  4.,  0.,  0.],
        [ 4.,  1.,  1.,  0.,  2., 92.,  0.,  0.],
        [ 0.,  0.,  1.,  2.,  0.,  0., 97.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0.,  1., 98.]])
```

Question 3
Steps for implementation:
1) At first we have to read images and labels for the data
2) Then build the resnet model and train the model and test the data.
3) Train accuracy is 93.2
4) Test accuracy is 89.32

ResNet-18 Architecture

Layer Name	Output Size	ResNet-18
conv-1	112 x 112 x 64	7 x 7, 64, stride 2
conv-2	56 x 56 x 64	3 x 3 max pool, stride 2 --- [3 x 3,64] X 2 [3 x 3,64]
conv-3	28 x 28 x 128	[3 x 3,128] X 2 [3 x 3,128]
conv-4	14 x 14 x 256	[3 x 3,256] X 2 [3 x 3,256]
conv-5	7 x 7 x 512	[3 x 3,512] X 2 [3 x 3,512]
Average pool	1 x 1 x 512	7 x 7 average pool
Fully connected	6	512 x 6 fully connections
softmax	6	

Parameters:
Total params: 11,179,590
Trainable params: 11,179,590
Non-trainable params: 0

Total Epocs :70

Loss and Accuracy:

Training Loss: 0.0058141967786823285
Training Accuracy: 94%
Validation Loss: 0.010291938092559577
Validation Accuracy: 89.375%

On cifar10 dataset:

Validation Accuracy -> 77.35%

Comparison:

With KNN

1)ResNet(DL Model)(with accuracy of around 89%) performs much better than the nearest neighbor classifier(ML Model)(with accuracy of around 49%) in terms of accuracy.
2)ResNet being a deep model was able to learn many complex features and therefore perform better as compared to the knn model, provided we have enough examples to train.
3)ResNet turned out to be the best according to the accuracy.

With Alexnet

Alexnet is a sequential model trained on Imagenet. So it may be the case that not many layers need to be changed as the features learnt for it are found to give high validation accuracy. bggResnet18 is a rather complex model. It seems to overfit into the data given the validation accuracy is much lesser than train accuracy. This particular happens when the model learns complex features unique to the training examples