CS-518 Computer Vision
Assignment02

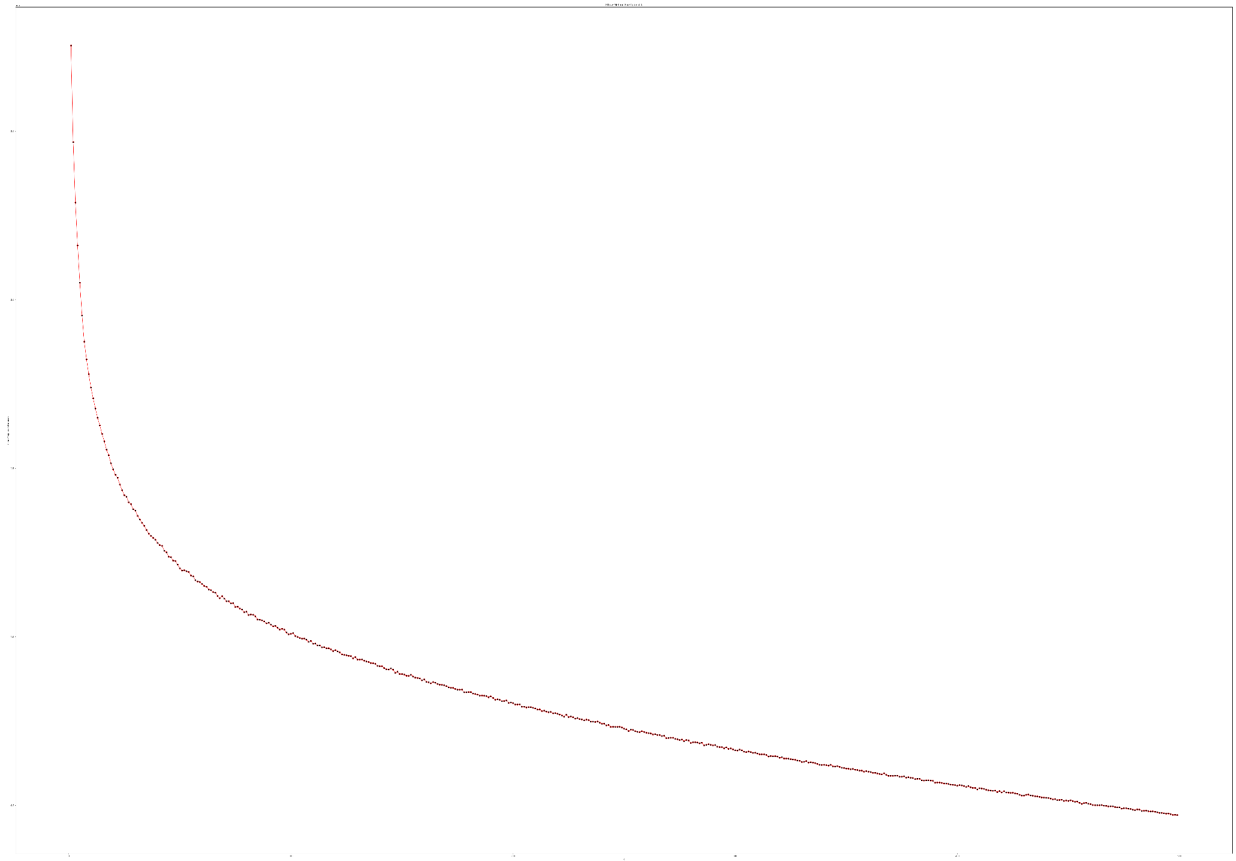
Bag Of Visual Words based image classification on MNIST dataset

Algorithm Steps:

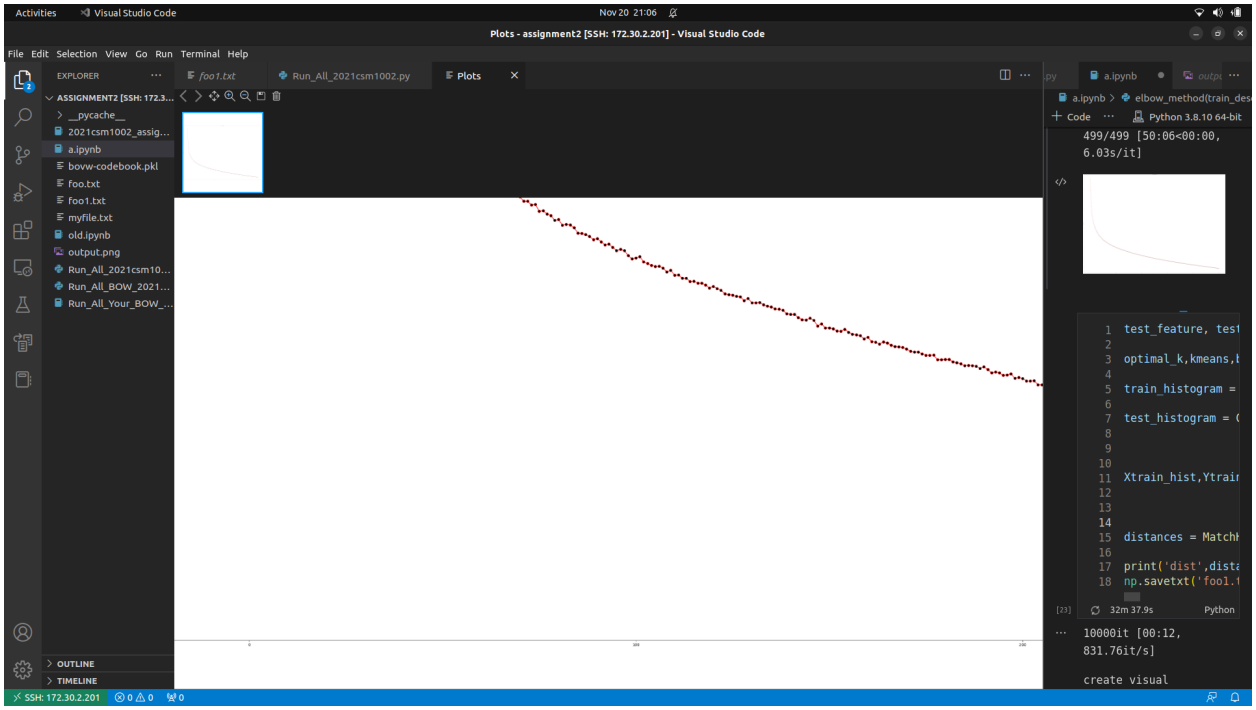
- **Dataset:** dataset is downloaded using tensorflow library which is divided into train and test.

- **Feature Extraction**: Use SIFT(Scale Invariant Feature Transform) to extract features such as keypoint and descriptors from train and test dataset.

- **KMeans Algorithm & selection of optimal k:** Apply Custom K Means algorithm to create a visual dictionary. The optimal value of k is decided using the **Elbow Method.** Train descriptors are sampled for finding the optimal value of k. The elbow method iterates over a range of 1 to 500 which computes the sum of squared distances which is the distance of point from its representation and inertia. The optimal value of k is decided by the point where the graph becomes linear or elbow point. For such a range of values of the optimal value of k is 200.



The experiment is conducted over different values of range such as up to 100 , 200 and 500. The observed elbow points are 40 , 120 and 200.It can be seen by zoom in the image opening in visual studio code. I have attached the screenshot of the zoomed version of image in current directory.



Saving Codebook and Distance from closest word: After applying the k-means , the generated cluster centers are saved as bovw_codebook,pkl in the current directory and the distance of closest centroid points during k-means algorithm are saved in file in the current directory.

Compute Histograms: Create histograms from train and test features. Then create Xtrain and Xtest histograms for performing the classification.

Match Histogram & Vector Quantization: Vector quantization basically saves the distances from k nearest cluster centers. Match histogram used train and test histograms along with its labels which use SVM classification and predict the accuracy along with classification report.

Results:

k=200

```
Accuracy = 0.6135846603834905
Confusion Matrix:
 [[596  24  53  67  32  35  43  18  58  18]
 [  2 455  10 108   9  12   4  11  17   0]
 [ 23  23 468  34 211  35  96  11  48   5]
 [ 96  48  80 434  59  37  31  45  81  12]
 [ 13  20 236  48 464  11 118   9  47   7]
 [ 10  16   7  29   1 749   6  96  13  39]
 [157  37 134  45 183  37 228  13  85  13]
 [  3  10  11  25   6  50   2 728  17 113]
 [ 15  38  58  43  31  33  23  22 688   7]
 [ 17   4   0  13   7  14   2  57  20 854]]
              precision    recall  f1-score   support

           0       0.64      0.63      0.64       944
           1       0.67      0.72      0.70       628
           2       0.44      0.49      0.47       954
           3       0.51      0.47      0.49       923
           4       0.46      0.48      0.47       973
           5       0.74      0.78      0.76       966
           6       0.41      0.24      0.31       932
           7       0.72      0.75      0.74       965
           8       0.64      0.72      0.68       958
           9       0.80      0.86      0.83       988
```

k=100

```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy = 0.5770772397356733
Confusion Matrix:
 [[583  22  52  68  43  59  24  15  68  10]
 [  4 451  21 101   8  18   1   5  18   1]
 [ 28  35 462  34 233  49  46   8  55   4]
 [104  67  91 381  61  54  15  41  95  14]
 [ 26  13 229  50 462  30  81   5  73   4]
 [ 15  27   4  36   4 713   5 103  14  45]
 [158  30 146  49 227  73 117  17 105  10]
 [ 16  11   9  36   5  52   1 678  20 137]
 [ 23  38  70  47  43  39  12  16 658  12]
 [ 23   6   3  13   6  40   0  46  29 822]]
              precision    recall  f1-score   support

           0       0.59      0.62      0.61       944
           1       0.64      0.72      0.68       628
           2       0.43      0.48      0.45       954
           3       0.47      0.41      0.44       923
           4       0.42      0.47      0.45       973
           5       0.63      0.74      0.68       966
           6       0.39      0.13      0.19       932
           7       0.73      0.70      0.71       965
           8       0.58      0.69      0.63       958
           9       0.78      0.83      0.80       988
...
```

k=70

```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy = 0.447188820279493
Confusion Matrix:
 [[528  45  34  58  37  66   5  39  92  40]
 [  1 456  14  83   9  23   1  23  16   2]
 [ 42  98 289  44 210  71  11  68 101  20]
 [104 185  75 267  61  46   9  42 101  33]
 [ 65  59 214  35 317  52  19  77 105  30]
 [ 34 119   9  36   6 569   0  58  38  97]
 [170  61 135  57 185  88  25  72 112  27]
 [ 36 101  12  64  19  52   1 382  55 243]
 [ 86  46  60  65  39  55  15  28 527  37]
 [ 41  11   5  28   3  40   0  42  50 768]]
              precision    recall  f1-score   support

           0       0.48      0.56      0.51       944
           1       0.39      0.73      0.50       628
           2       0.34      0.30      0.32       954
           3       0.36      0.29      0.32       923
           4       0.36      0.33      0.34       973
           5       0.54      0.59      0.56       966
           6       0.29      0.03      0.05       932
           7       0.46      0.40      0.43       965
           8       0.44      0.55      0.49       958
           9       0.59      0.78      0.67       988
```

k=400 (builtin k means)

```
Output exceeds the size limit. Open the full output data in a text editor
Accuracy = 0.6456505254035316
Confusion Matrix:
 [[623  17  49  76  30  30  58  12  40   9]
 [  9 481  11  92   7   8   1   4  14   1]
 [ 26  17 527  37 191  21  85   3  45   2]
 [ 96  48  76 452  69  40  42  36  59   5]
 [ 13   8 207  43 498  14 130   6  50   4]
 [  4  13   4  18   3 780   6  98  12  28]
 [152  32 147  51 163  32 258   3  88   6]
 [  2   6   4  22   3  51   2 759  15 101]
 [ 20  21  55  47  25  22  28  16 717   7]
 [  7   3   4   9   1  16   3  61  19 865]]
              precision    recall  f1-score   support

           0       0.65      0.66      0.66       944
           1       0.74      0.77      0.76       628
           2       0.49      0.55      0.52       954
           3       0.53      0.49      0.51       923
           4       0.50      0.51      0.51       973
           5       0.77      0.81      0.79       966
           6       0.42      0.28      0.33       932
           7       0.76      0.79      0.77       965
           8       0.68      0.75      0.71       958
           9       0.84      0.88      0.86       988
...
```