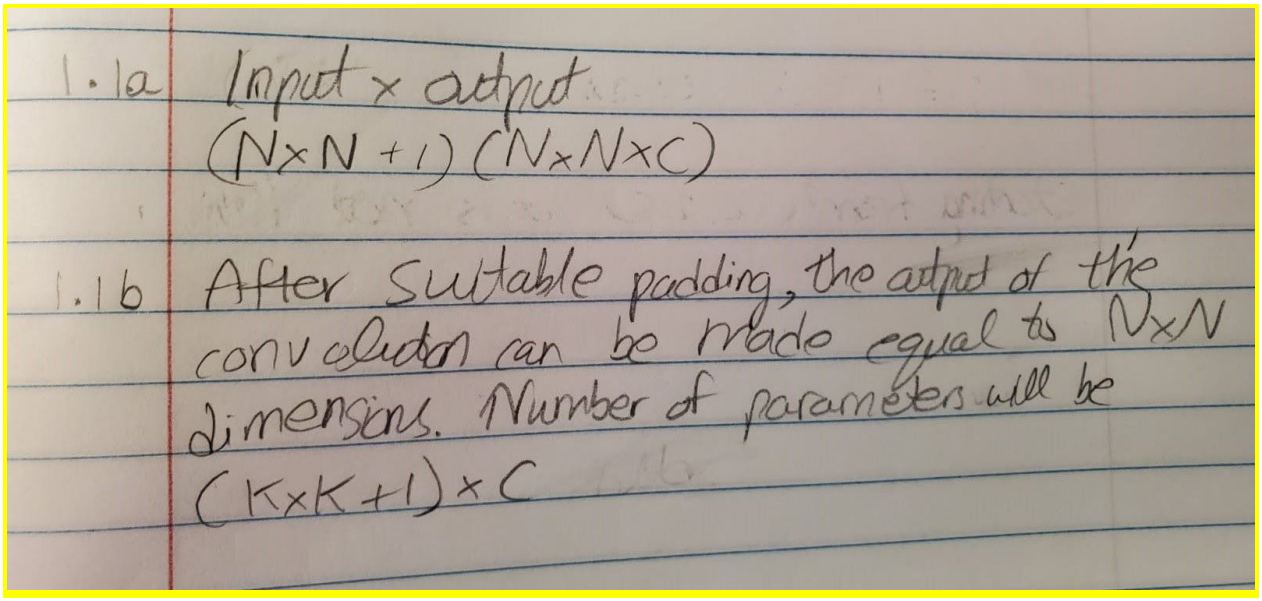
Umair Afzal
HW5
CS589

Collaborators:
Debankita,
Bill

1. Convolutional Neural Networks [40 points]

1.1.



1.2.

Output from CNN_Template.py

```
Train Epoch: 0  Loss: 2.297700
Train Epoch: 0  Loss: 1.196974
Train Epoch: 0  Loss: 0.879102
Train Epoch: 0  Loss: 0.631757
Train set Accuracy: 93.1%
Test set Accuracy: 92.8%

Train Epoch: 1  Loss: 0.489871
Train Epoch: 1  Loss: 0.481325
Train Epoch: 1  Loss: 0.555330
Train Epoch: 1  Loss: 0.275723
Train set Accuracy: 95.5%
Test set Accuracy: 95.0%

Train Epoch: 2  Loss: 0.295671
Train Epoch: 2  Loss: 0.565036
Train Epoch: 2  Loss: 0.677427
```

Train Epoch: 2 Loss: 0.511953
Train set Accuracy: 96.6%
Test set Accuracy: 95.7%

Train Epoch: 3 Loss: 0.419044
Train Epoch: 3 Loss: 0.362855
Train Epoch: 3 Loss: 0.424028
Train Epoch: 3 Loss: 0.605251
Train set Accuracy: 97.0%
Test set Accuracy: 96.2%

Train Epoch: 4 Loss: 0.290133
Train Epoch: 4 Loss: 0.524181
Train Epoch: 4 Loss: 0.262745
Train Epoch: 4 Loss: 0.562951
Train set Accuracy: 96.8%
Test set Accuracy: 95.7%

Train Epoch: 5 Loss: 0.470912
Train Epoch: 5 Loss: 0.643341
Train Epoch: 5 Loss: 0.385698
Train Epoch: 5 Loss: 0.815582
Train set Accuracy: 97.9%
Test set Accuracy: 96.5%

Train Epoch: 6 Loss: 0.110166
Train Epoch: 6 Loss: 0.386724
Train Epoch: 6 Loss: 0.626907
Train Epoch: 6 Loss: 0.300639
Train set Accuracy: 97.9%
Test set Accuracy: 96.6%

Train Epoch: 7 Loss: 0.245308
Train Epoch: 7 Loss: 0.248199
Train Epoch: 7 Loss: 0.602157
Train Epoch: 7 Loss: 0.261321
Train set Accuracy: 98.2%
Test set Accuracy: 96.7%

Train Epoch: 8 Loss: 0.263233
Train Epoch: 8 Loss: 0.242980
Train Epoch: 8 Loss: 0.320146
Train Epoch: 8 Loss: 0.397581

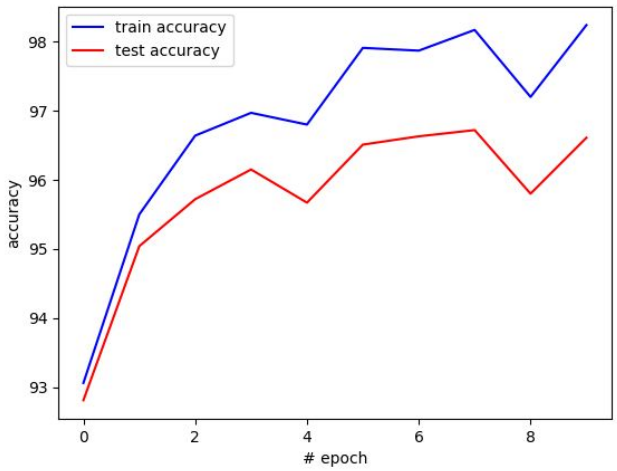
Train set Accuracy: 97.2%
Test set Accuracy: 95.8%

Train Epoch: 9 Loss: 0.449104
Train Epoch: 9 Loss: 0.264772
Train Epoch: 9 Loss: 0.233881
Train Epoch: 9 Loss: 0.552281
Train set Accuracy: 98.2%
Test set Accuracy: 96.6%

1.2.a

o/p size	**#Params**
[26x26x32]	**320**
[24x24x64]	**18496**
[12x12x64]	**0**
[12x12x64]	**0**
[9216]	**0**
[128]	**1179776**
[128]	**0**
[10]	**1290**

1.2.b



1.2.c

2.c Params in convolutional layer = 320 + 18496 = 18816

params in fully connected (FCL) layer = 1179776 + 1290 = 1181066

Fraction of conv = $\frac{18816}{1181066 + 18816} = 0.015681542$

fraction of FCL = $\frac{1181066}{1199882} = 0.984318458$

Not surprising. As we know that a convolution layer having kernel = dimension gives us a fully connected layer. We reduce the dimensions of image in convolution due to which we end up having minimum weight and memory as compared to fully connected layer.

2. Singular Value Decomposition [40 points + 10 Extra Credit]

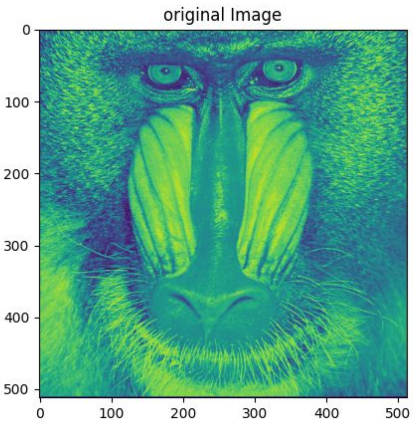
2.1

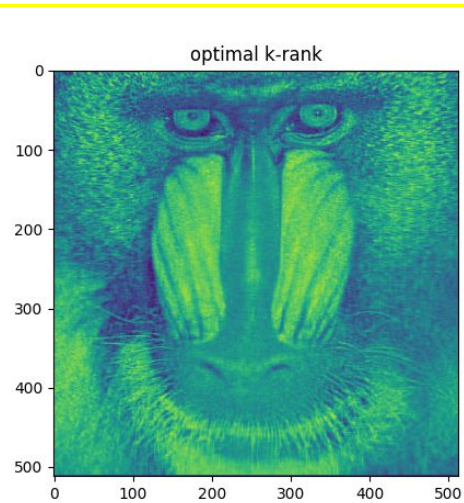
```
def SVD(A, s, k):
    # TODO: Calculate probabilities p_i
    n,m = A.shape
    probabilities = list()
    counter = 0
    while counter < n:
        upper = math.pow(np.linalg.norm(A[counter,:], ord=2), 2)
        lower = math.pow(np.linalg.norm(A), 2)
        probabilities.append(upper / lower)
        counter += 1
    # TODO: Construct S matrix of size s by m
    S = np.zeros((s,m))
    counter = 0
    while counter < s:
        random_choice = np.random.choice(list(range(n)), p=probabilities)
        S[counter] = A[random_choice,:]
        counter += 1

    # TODO: Calculate SS^T
    ss_to_t = np.matmul(S, S.T)

    # TODO: Compute SVD for SS^T
    svd = np.linalg.svd(ss_to_t)
    W = svd[0]
    values = svd[1]

    # TODO: Construct H matrix of size m by k
    H = np.zeros((k, m))
    counter = 0
    while counter < k:
        mat_mul = np.matmul(S.T, W[:, counter])
        linalg_norm = np.linalg.norm(mat_mul, ord=2)
        H[counter] = mat_mul / linalg_norm
        counter += 1
    values.sort()
    values = values[::-1]
    top_k_singular_values_sigma = values[:k]

    # Return matrix H and top-k singular values sigma
    return H.T, top_k_singular_values_sigma
```

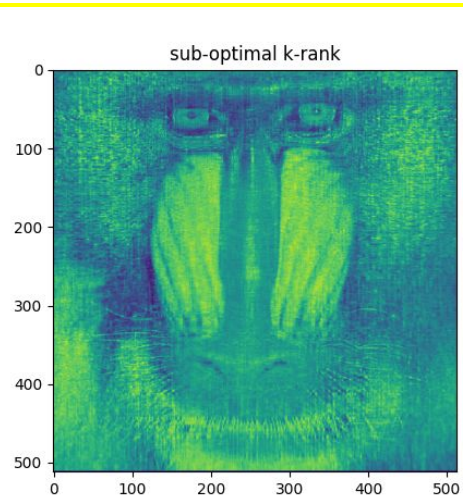
2.2 EXTRA CREDIT

2.3

2.3a







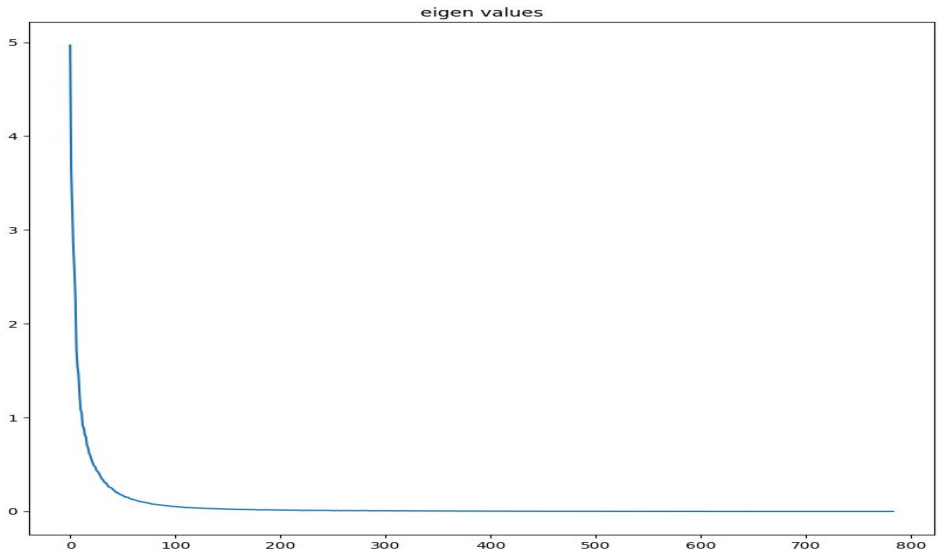
2.3b

8827.532897298843
11399.808798648595

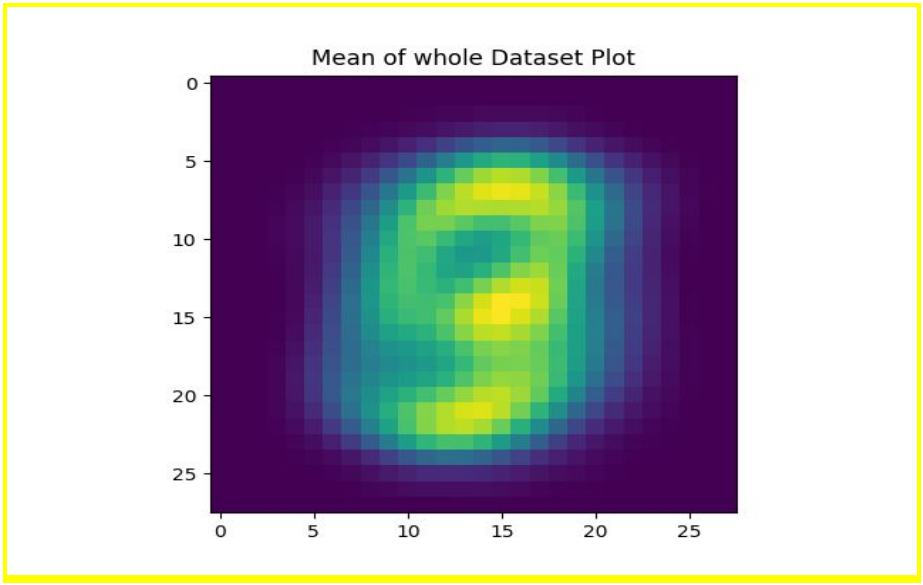
3. Principle Component Analysis [20 points]

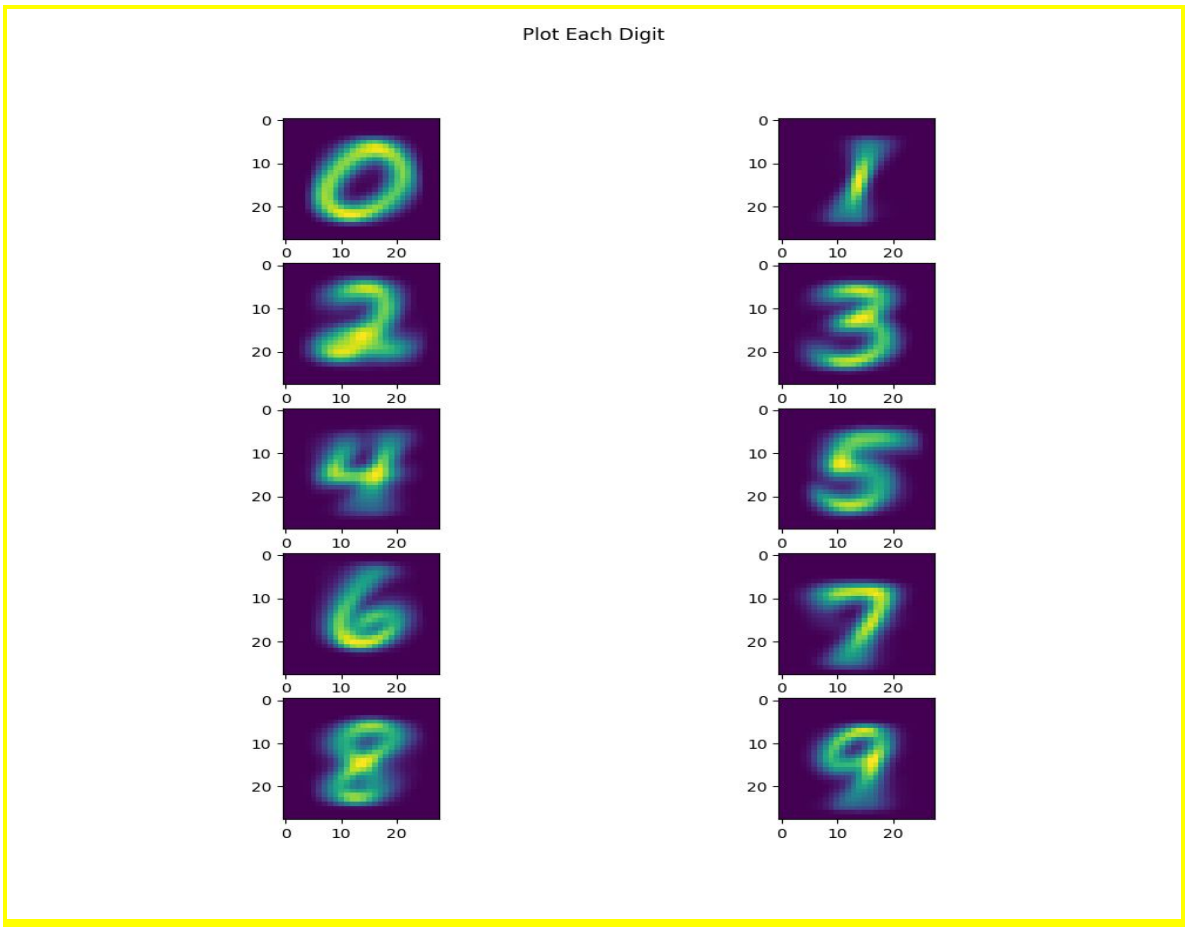
3.1

I will use 30 because after that the eigenvalues were insignificant.

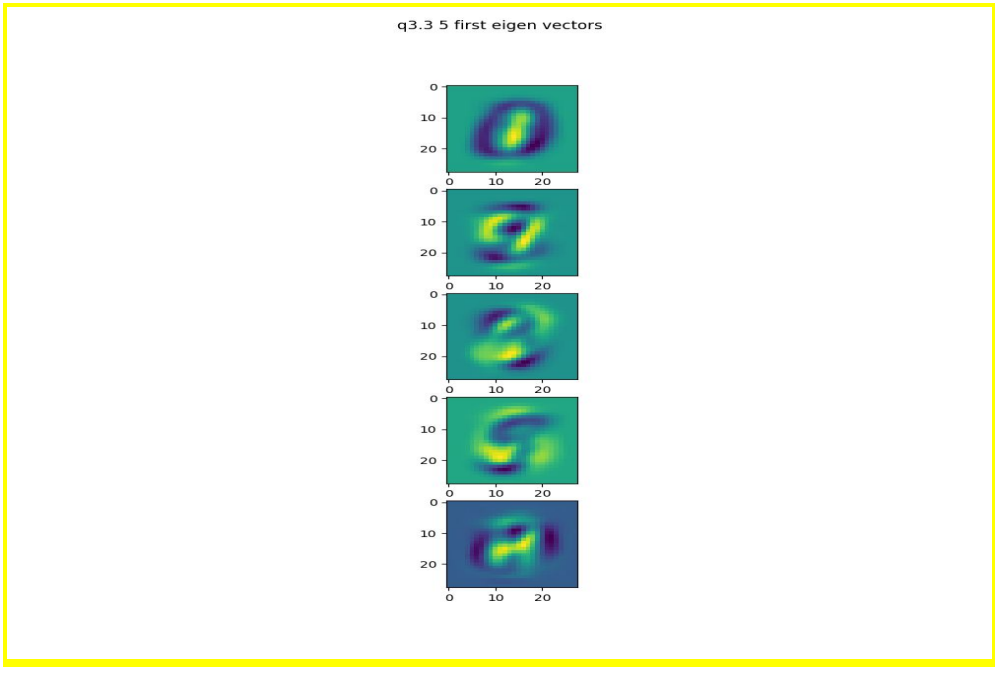


3.2





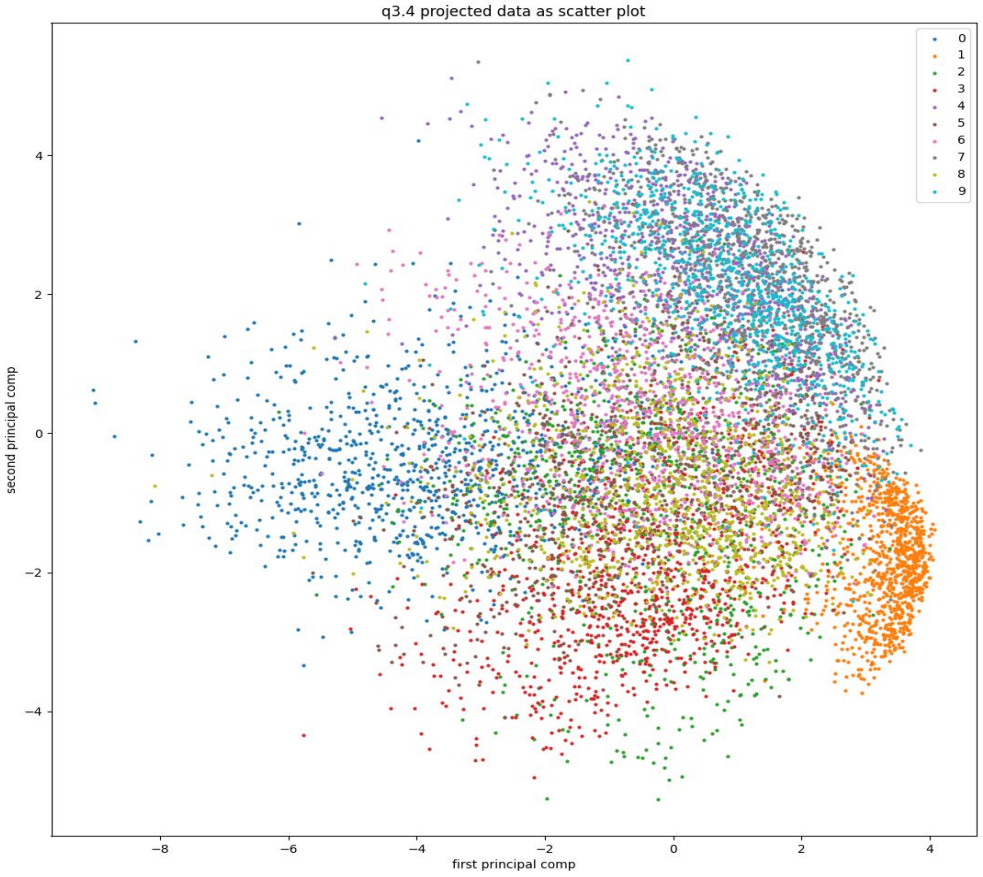
3.3



0, 9, 3, 5, 2

3.4

Digit 7 and 9 are close to each other. The number 8 is scattered in the center as it has similarity to most of the digits



Sources

https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/