LAB REPORT

Aman Bhansali(B20ME010)

TASK 1:

1.1

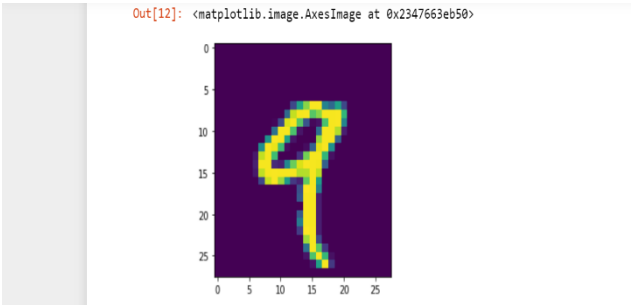
Downloaded the mnist dataset.

1.2

This dataset contains pixel values corresponding to a particular digit. The size of the initial mnist dataset is 70000, 785.

1.3

I made a variable named features in which I stored the pixel values from the original mnist dataset and a variable named labels which stored the digits. Then I converted my dataframe of features to numpy array and used its fourth index which means the first image and then reshaped the 784 data to (28, 28). Then used the matplotlib's imshow to see the image.




1.4

Then used the inbuilt function of python of PCA and gave any random number of components value and then fitted the first 5000 rows/images to the pca and then using the library function calculate the variance contributed by each component and the total variance by dimensionality reduction.

Then we need to give a particular value of variance we wanted and then observe the corresponding number of components needed for it. For this I gave the value equal to 0.94 and got the required number components equal to 134.

All these values calculated were using the library function of PCA.

Colab link Task 1 : https://drive.google.com/file/d/1vZbsbpm3HuxfjjpCX93-xRjOy6-8RLSY/view?usp=sharing

References: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Task 2

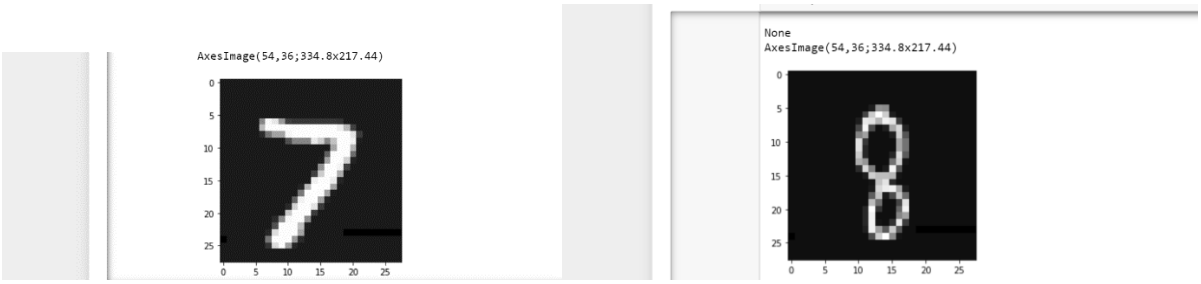
2.1 Downloaded the mnist dataset. Using pandas added the dataset to my notebook. Also imported libraries such as numpy, matplotlib. This dataset contains pixel values corresponding to a particular digit. The size of the initial mnist dataset is 70000, 785. I made a variable named features in which I stored the pixel values from the original mnist dataset and a variable named labels which stored the digits. Then made a new variable and stored the first 5000 samples of features in it.

Then using the function of .cov() I calculated the covariance matrix of the new features dataframe containing 5000 samples. The shape of covariance matrix I got was (784, 784).

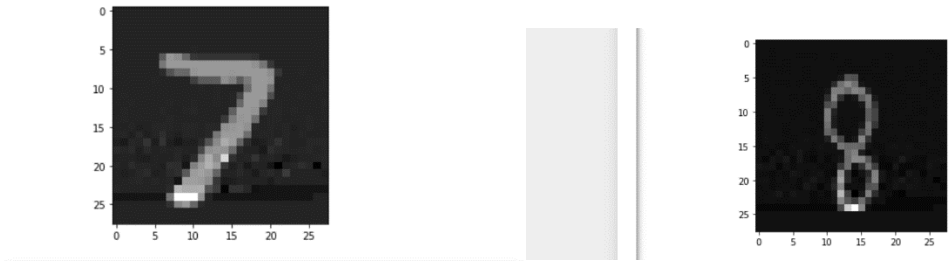
2.2 Then we needed to give the eigen vectors and eigen values from the covari ance matrix. For this I used numpy's linalg to calculate the eigen values and eig en vectors from the covariance matrix. Now the values we got were not arrang ed as we wanted the values in descending order for this I used the arg sort fun ction which is used to arrange in ascending order. We needed the same but in descending order so to do that I used indexing and gave -1 value to reverse tha t and stored the index as per descending order of eigen values. The first five eig en values were: 337829.65625394 253504.78675022 209790.38118221 18505 8.52464273, 162197.10888392
Now to arrange the eigen vectors also in descending orderI used the same Indexing.

2.3
We needed to reconstruct the images now taking values of components equal to 10, 50, 100, 300, 700.
Then I made a tuple which would contain all the values of components. Then used for loop to iterate over the component values stored in the tuple and taki ng each principal component I used indexing and took the number of eigen vectors as per the value. Also while calculating the eigen values and vectors they got converted into complex so to convert them to real I used .real

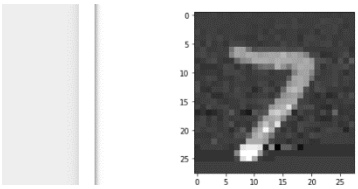
function. After this I made a variable data_ which stored the value of original data(converted into numpy) – mean for the corresponding rows for each values wich is something like origin shift. Then as we needed to implement PCA from scratch I took dot product of the transpose of eigen vecto rs(no. equal to the no. of components like 10, 50..) with the mean subtracted data and took the overall transpose which gave an array of size = 5000, 10 After this I again took a dot product of my new dot product with the eigen vect ors and got a shape of 5000, 784 and then added the mean subtracted data again to it to make it of original size so that while plotting we have same 28,28 values with us. Now we had the array which contained the values corresponding to the number of components. Then I just did the same as reshaped my array to 28, 28 and then plotted the image for all the components. The values I used were 4000, 4006 index.
The reconstructed images were :
For n=10

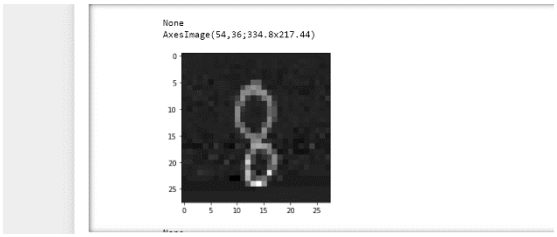



For n=50

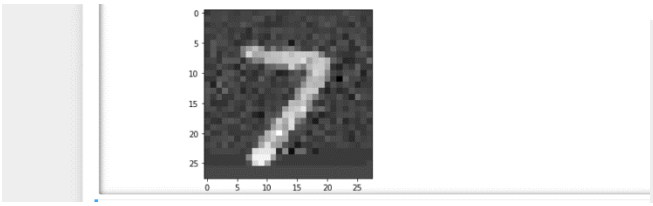


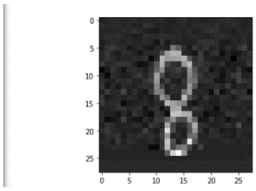
For n=100



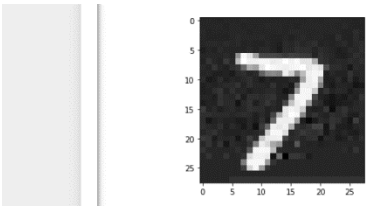


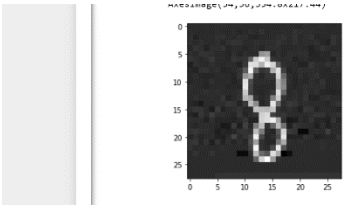

For n=300





For n=700

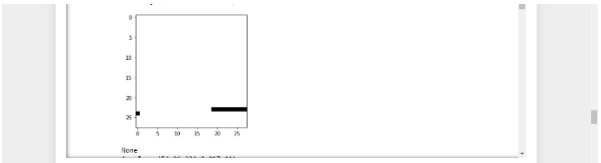


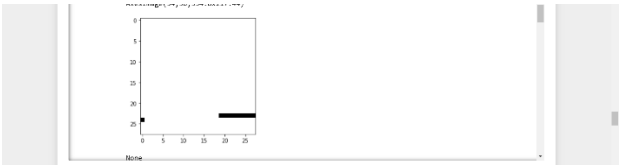


2.5

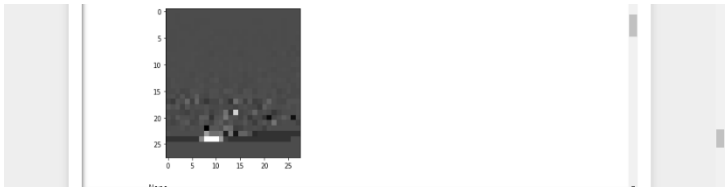
For residual image just subtracted the reconstructed from the original 28, 28.

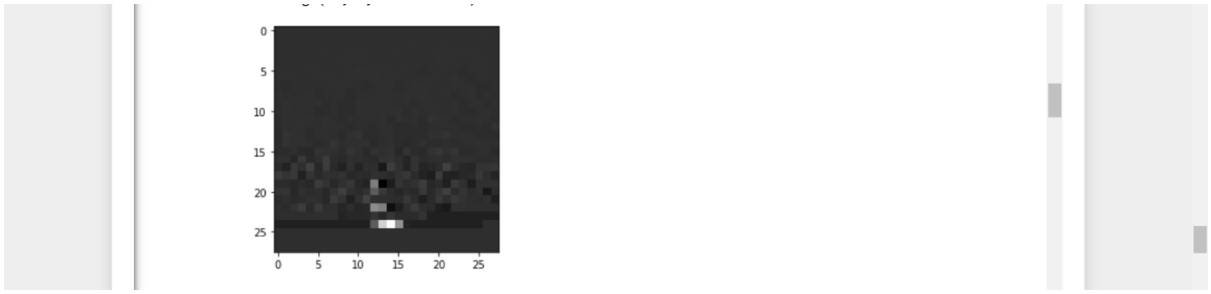
For n=10



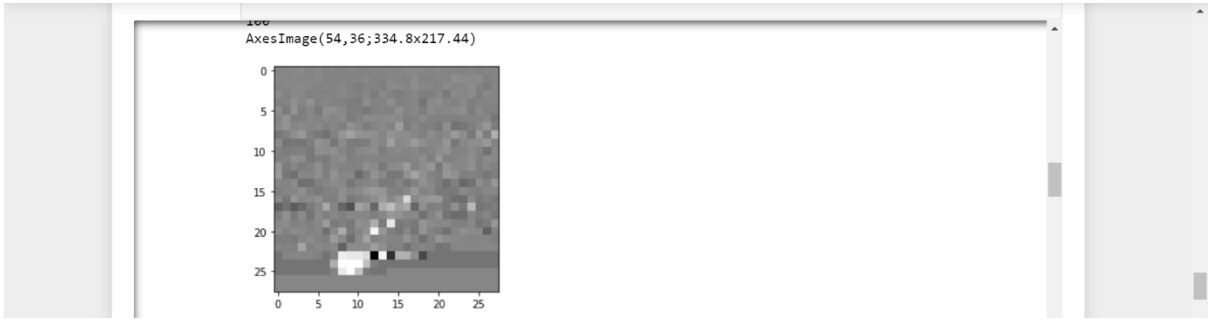


For n=50



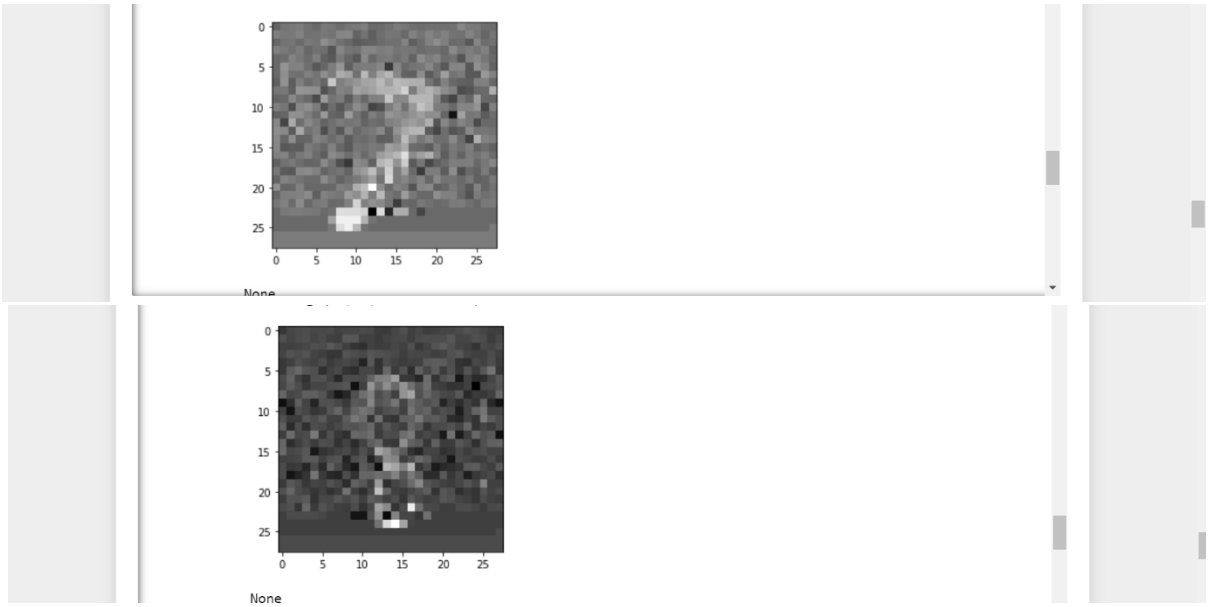


For n=100



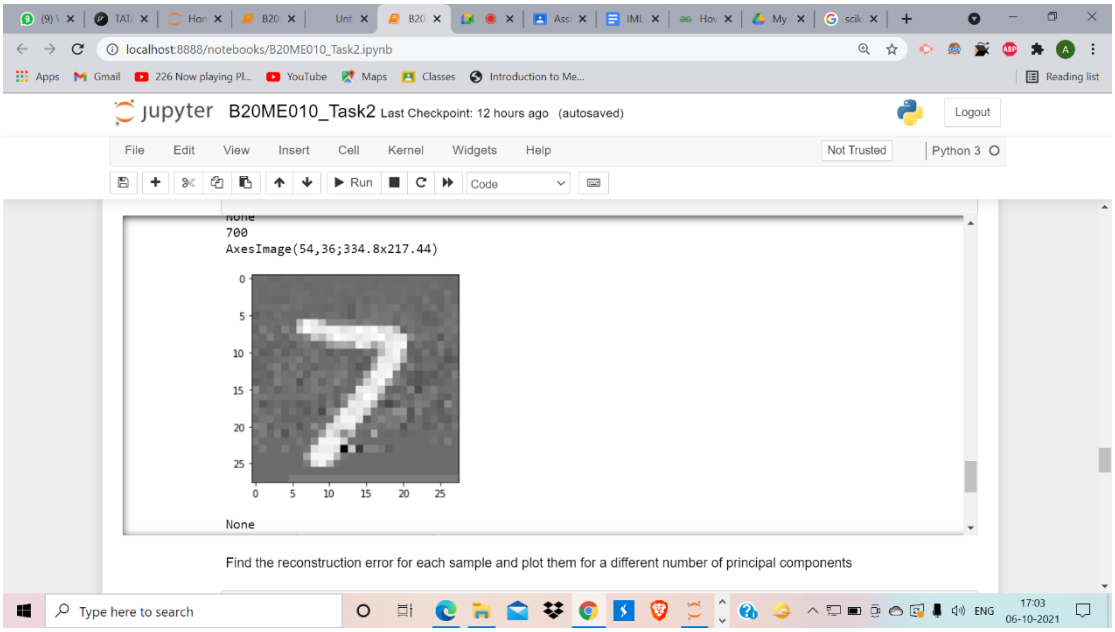


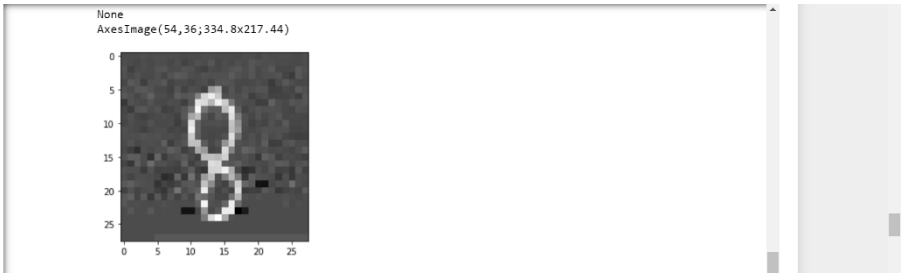
For n=300


None


None

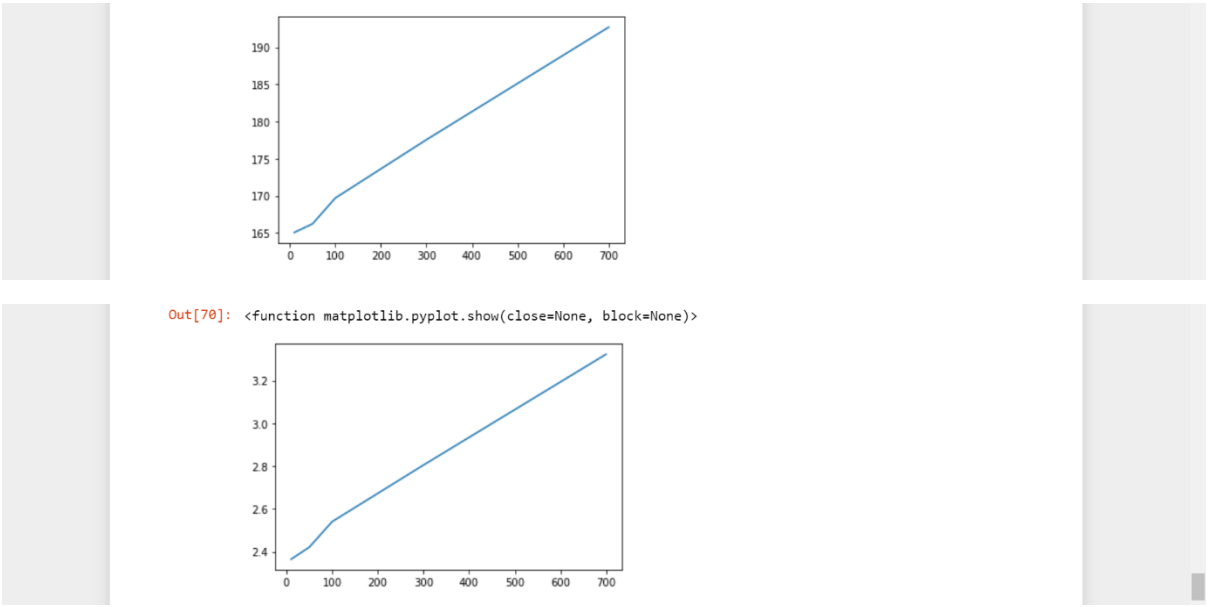
For n=700


jupyter B20ME010_Task2 Last Checkpoint: 12 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
700
AxesImage(54,36;334.8x217.44)
None
Find the reconstruction error for each sample and plot them for a different number of principal components
Type here to search
ENG
17:03
06-10-2021


None
AxesImage(54,36;334.8x217.44)

2.6

To calculate the rms just flattened the residual array for each case and then took the squared values and stored it. Then divided by 784 and used the math.sqrt to obtain the rms error.




Task 2 link
https://drive.google.com/file/d/1L8Cbax9y1eav1HTMKxS9CcYw6iIWe0kr/view?usp=sharing

References: https://towardsdatascience.com/principal-component-analysis-pca-from-scratch-in-python-7f3e2a540c51

https://stackoverflow.com/questions/13224362/principal-component-analysis-pca-in-python

https://datahub.io/machine-learning/mnist_784#resource-mnist_784