CSL7670 : Fundamentals of Machine Learning

Lab Report

Name:	**SOUHITYA KUNDU**
Roll Number:	**M20PH209**
Program:	**MSc-MTech(Physics & Materials Sc.)**

Chapter 1

Lab-10

1.1 Objective

The objective of this whole assignment is to learn about PCA(Principle Component Analysis)

1.2 Problem-1

The main objective is to understand how the Dimensionality reduction takes place using the PCA algorithm.

- I have tried here to play with the K values and understand the effect of the parameter on reducing the dimensions and still retrieving the maximum information out of the images.

Solution 1:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:



# a) , b)
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import fetch_openml

# Load the MNIST dataset (or any other suitable dataset)
mnist = fetch_openml('mnist_784')
X = mnist.data.astype('float64') / 255.0  # Normalize pixel values to [0,
    ↪ 1]


# In[ ]:



# Number of principal components to retain
# n_components = 20

n_comp = [2,4,8,10,30,40,50,75,80,90,100]

for n in n_comp:
```

```
    # Perform PCA
    pca = PCA(n_components=n)
    X_pca = pca.fit_transform(X)



    # Reconstruct the data
    X_reconstructed = pca.inverse_transform(X_pca)

    # Reshape the images for plotting
    original_images = X.values.reshape((-1, 28, 28))
    reconstructed_images = X_reconstructed.reshape((-1, 28, 28))

    # Plot original and reconstructed images
    def plot_gallery(images, n_row=5, n_col=5):
        plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
        plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace
            ↪ =.35)
        for i in range(n_row * n_col):
            plt.subplot(n_row, n_col, i + 1)
            plt.imshow(images[i], cmap=plt.cm.gray)
            plt.title(f'Digit␣{i}')
            plt.xticks(())
            plt.yticks(())

    # Display original images
    plot_gallery(original_images, n_row=1, n_col=5)
    plt.suptitle('Original␣Images', size=16)
    plt.show()

    # Display reconstructed images
    plot_gallery(reconstructed_images, n_row=1, n_col=5)
    plt.suptitle(f'Reconstructed␣Images␣(n_components={n})', size=16)
    plt.show()



# In[ ]:


# C)
import numpy as np

recon_loss = []

for n in n_comp:
    # Perform PCA
    pca = PCA(n_components=n)
    X_pca = pca.fit_transform(X)
    X_recon = pca.inverse_transform(X_pca)
    loss_vals = np.mean((X - X_recon)**2)
    recon_loss.append(loss_vals)

    # Plot the reconstruction losses against different values of N
```

```
plt.plot(n_comp, recon_loss)
plt.title('Reconstruction Loss Values vs. Principal Components (N)')
plt.xlabel('Principle Component Numbers (N)')
plt.ylabel('Reconstruction Loss Values')
plt.show()







# In[ ]:


X_pca


# In[ ]:
```

We witness that as we reduce the Dimension,but still go till 100th K value,we get quite a good accuracy almost near to the original image.

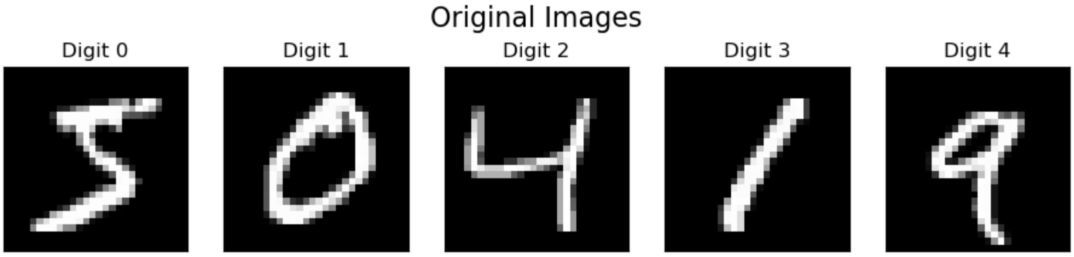

Figure 1.1: The Original Image

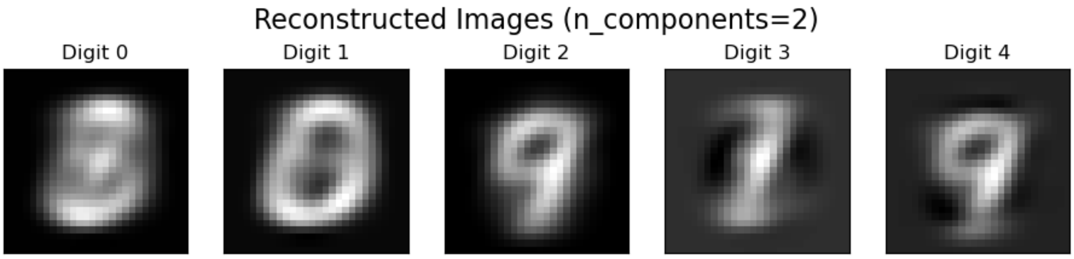

Figure 1.2: Dimension Reduced Image

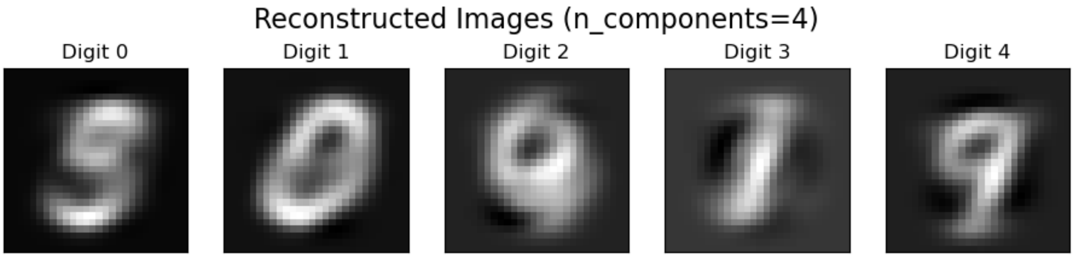

Figure 1.3: Dimension Reduced Image

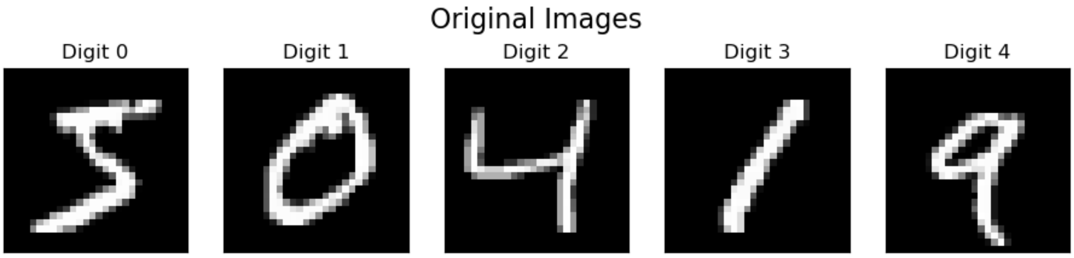

Figure 1.4: Dimension Reduced Image

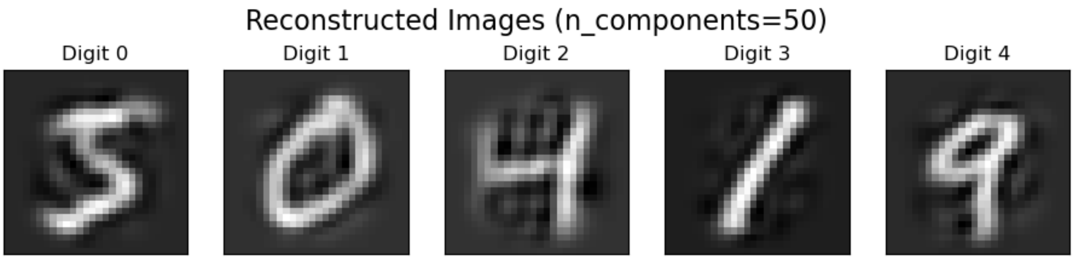

Figure 1.5: Dimension Reduced Image

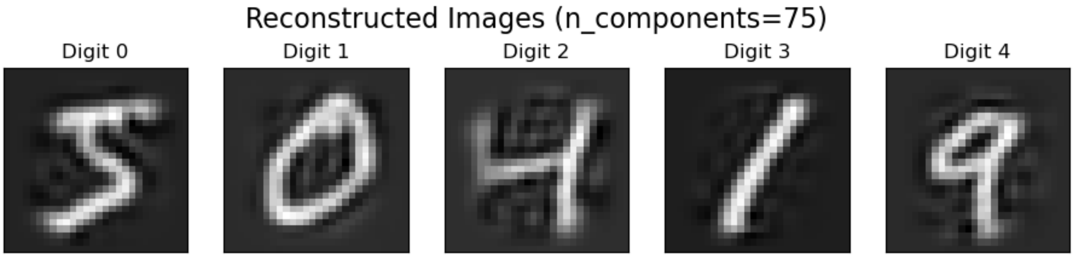

Figure 1.6: Dimension Reduced Image

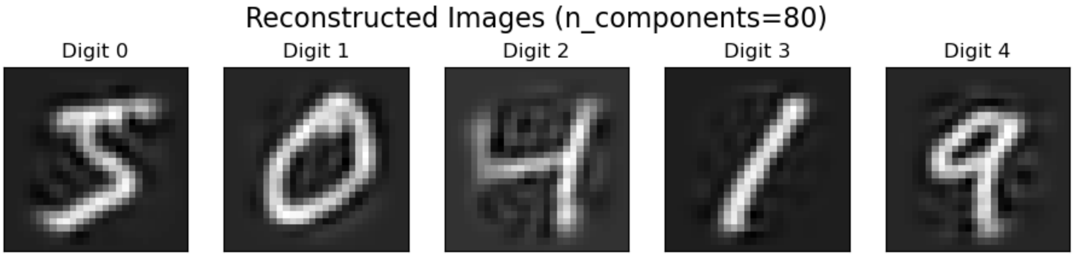

Figure 1.7: Dimension Reduced Image

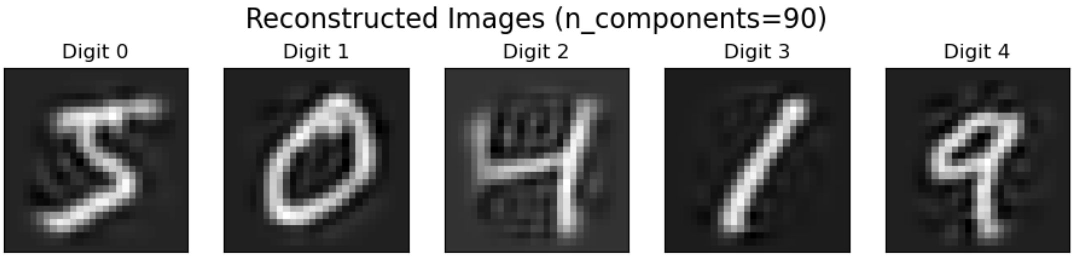

Figure 1.8: Dimension Reduced Image

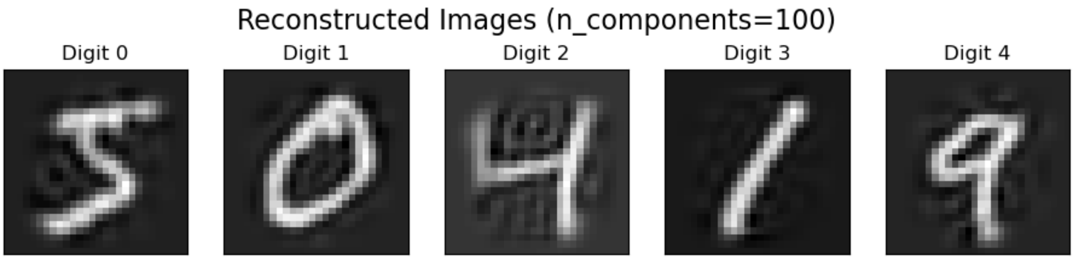

Figure 1.9: Dimension Reduced Image

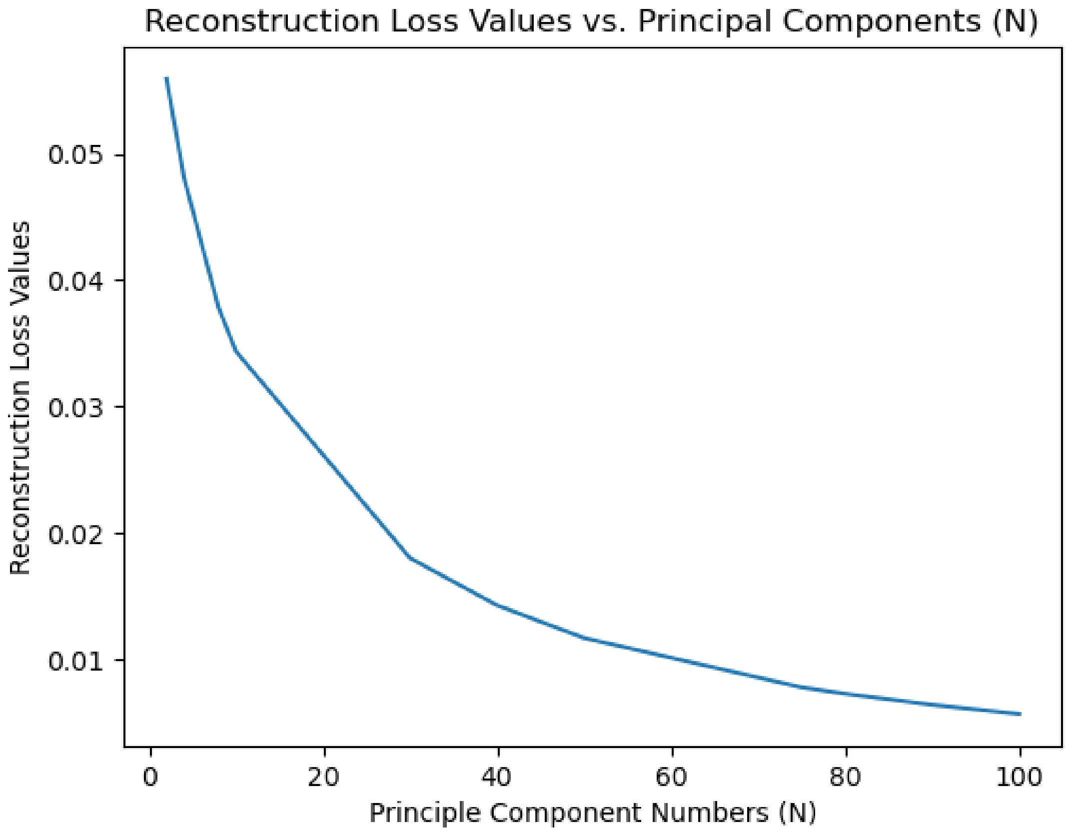

Figure 1.10: The Loss function plot