**Machine Learning Course**

# Noise Reduction Using PCA and Autoencoder Assignment Report

**Professor:**
Dr. Mahdi Eftekhari

**Author:**
Amirhossein Abolhassani

**Fall 2024**

## Contents

## 1 Introduction

One of the tasks in image processing is Noise Reduction. This task aims to reduce image noise using various algorithms and techniques. Two fundamental techniques are explored in this assignment and tested on the Fashion MNIST dataset:

- Principal Component Analysis (PCA)
- Autoencoder

## 2 Dataset Description

The Fashion MNIST dataset is widely used in machine learning and computer vision. It consists of grayscale images of various clothing and accessories, serving as a modern alternative to the classic MNIST dataset. A sample of the data is shown in Figure 1.

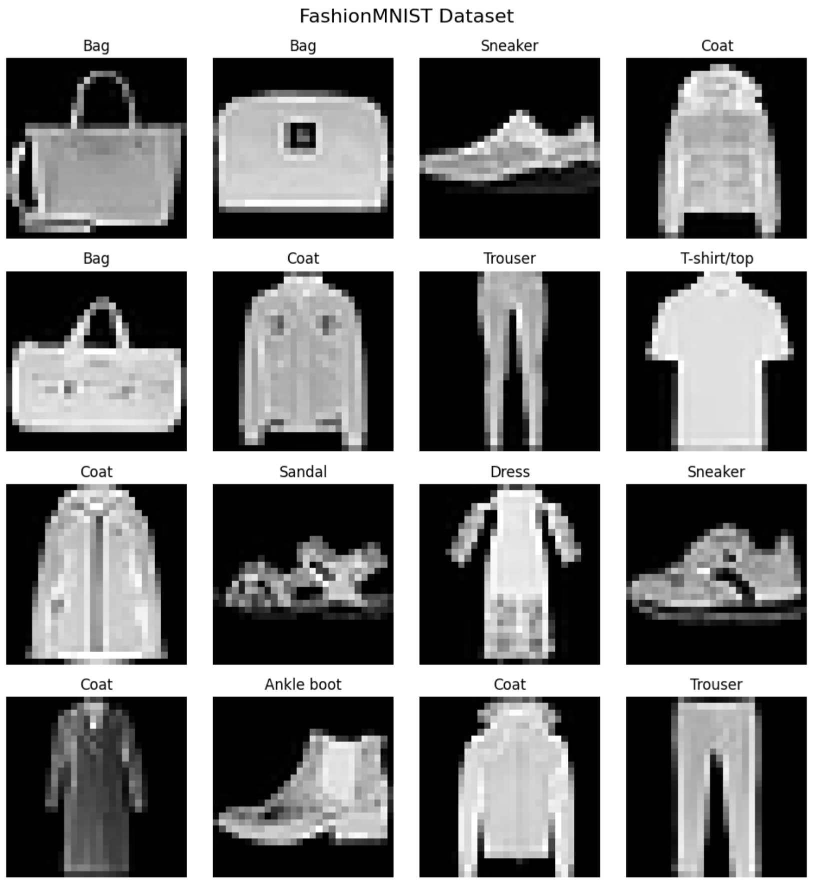

Figure 1: Subset of the Fashion MNISTdataset

| | |
|---|---|
| Total number of images | 70,000 |
| Training set | 60,000 |
| Test set | 10,000 |
| Image dimensions | 28×28 pixels |
| Image type | Grayscale (single-channel) |
| Pixel value range | 0 to 255 |

Table 1: Characteristics of the Fashion MNIST dataset

The dataset includes 10 distinct classes:

- T-shirt/top
- Trousers
- Pullover
- Dress
- Coat
- Sandal
- Shirt
- Sneaker
- Bag
- Ankle boot

## 3 Adding Gaussian Noise to the Dataset

Gaussian Noise is a type of noise added to images. The following probability density function describes this noise in two dimensions, as applied to images:

$$n(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}$$

| | |
|---|---|
| $\mu$ | Mean |
| $\sigma$ | Standard deviation, indicating noise spread |
| $\sigma^2$ | Variance |

In image processing, the formula for adding Gaussian noise to an image is:

$$g(x, y) = f(x, y) + n(x, y)$$

| | |
|---|---|
| $g(x, y)$ | Noisy image |
| $f(x, y)$ | Original image |
| $n(x, y)$ | Gaussian noise following a normal distribution with specified mean and standard deviation |

Initially, Gaussian noise with a variance of 1.0 and mean of 0 is added to the Fashion MNIST-dataset. These noisy images are used throughout the assignment.[1] (A subset of the noisy dataset is shown in Figure 2.)

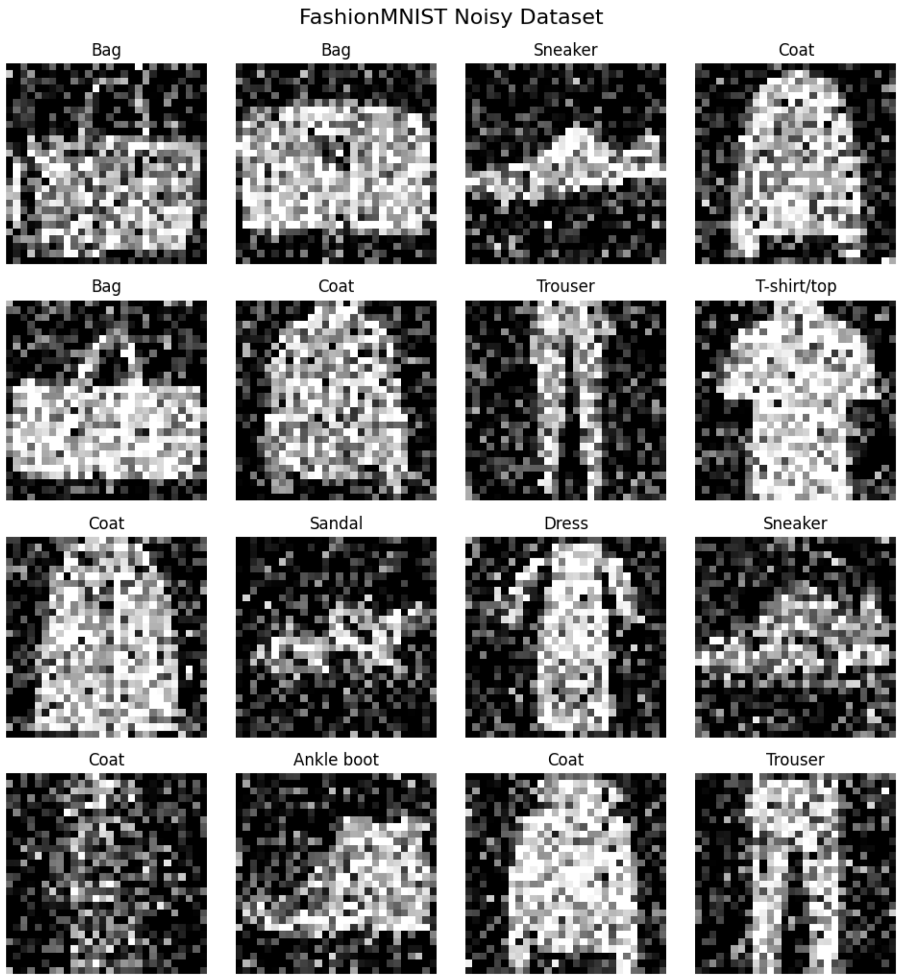

Figure 2: Subset of the noisy Fashion MNISTdataset

[1]In practice, each image is converted to a vector of length 784 for use in this assignment.

## 4 Noise Reduction with PCA

PCA, as a linear dimensionality reduction method, preserves important data components. By treating noise as less significant information, PCA is expected to reduce noise in images. The process involves computing the matrix $Q^1$, which includes the eigenvectors of the covariance matrix $\Sigma$:

$$\Sigma = Q \Lambda Q^{-1}$$

$$Q = [q_1 \; q_2 \; \cdots \; q_d]$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

Then, the data is projected onto the principal components:

$$Y = (X - \mu)Q$$

Finally, the data is reconstructed to its original dimensionality:

$$X' = YQ^T + \mu$$

By selecting an appropriate number of principal components, this dimensionality reduction and reconstruction process is expected to remove significant noise from the image.
In this assignment, 80 principal components yielded the best result among 784 possible components, achieving a reconstruction error of 0.0239, the lowest among tested values (Figure 3).

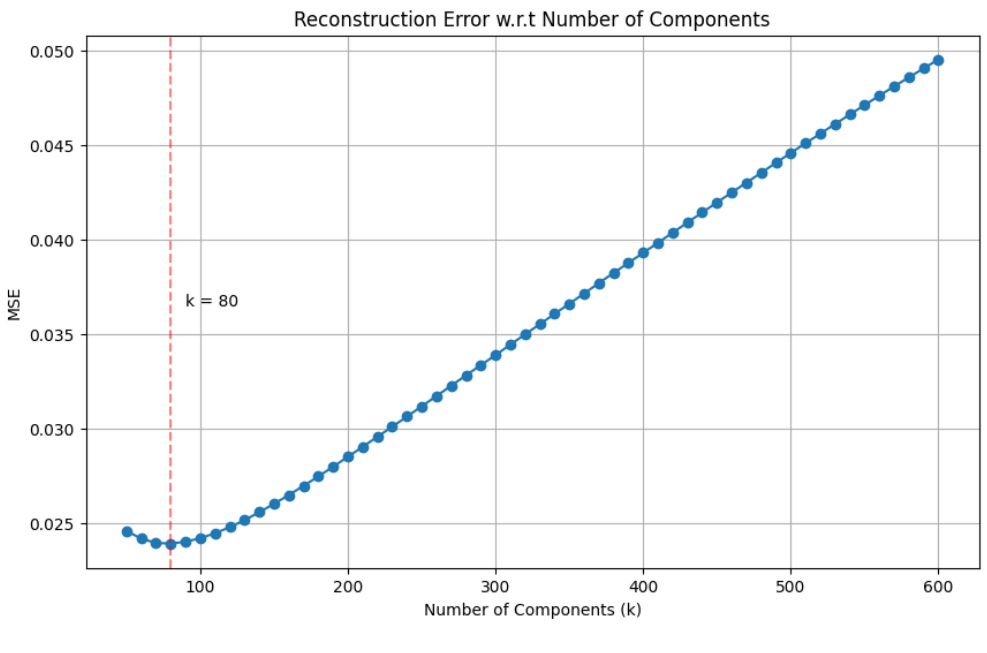

Figure 3: Finding the optimal number of components based on reconstruction error

[1]The matrix $Q$ contains orthogonal unit vectors, so $Q^{-1} = Q^T$.

## 5 Noise Reduction with Autoencoder

An Autoencoder is a neural network consisting of an Encoder and a Decoder. It can perform linear (with linear activation functions) or nonlinear (with nonlinear activation functions) dimensionality reduction.

One application of this architecture is to input noisy images and use the corresponding clean images as labels, training the Autoencoder to map noisy images to clean ones, which is the desired model.

A PyTorch-based network with Dense layers and ReLU activation functions was implemented. The layer structure is as follows:

$$\text{Encoder} : 784 \rightarrow 512 \rightarrow 256 \rightarrow 128$$

$$\text{Decoder} : 128 \rightarrow 256 \rightarrow 512 \rightarrow 784$$

Figure 4 shows the training and test error trends, indicating proper learning without overfitting. With the hyperparameters listed in Table 2, an error of 0.013 was achieved.

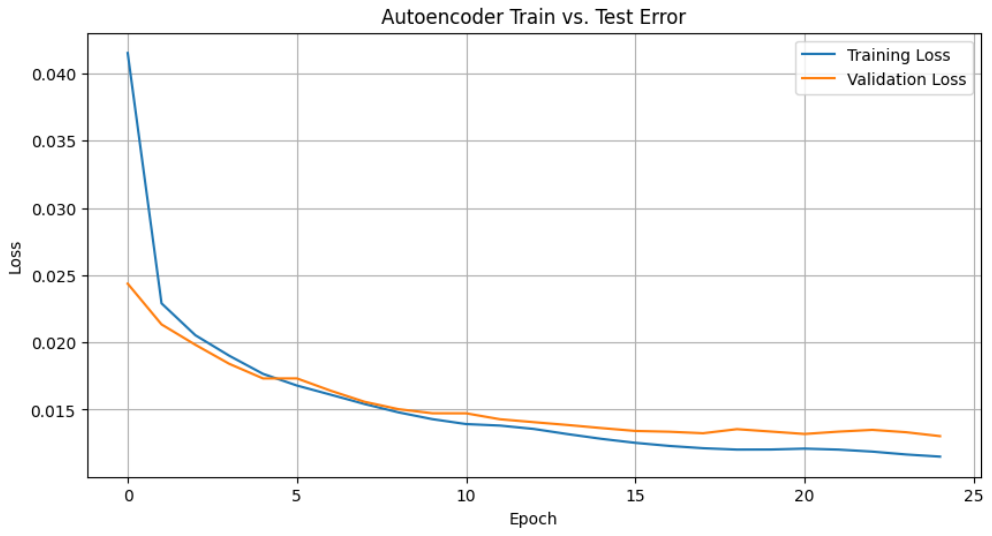

Figure 4: Training and test error trends for the Autoencoder

| | |
|---|---|
| Learning rate | 0.001 |
| Batch size | 128 |
| Epochs | 25 |

Table 2: Hyperparameters for the Autoencoder

## 6 Results

Comparing the errors reported in Sections 5 and 4, the Autoencoder is expected to outperform PCA in noise reduction. Figure 5 compares the noisy image, original image, PCA-denoised image, and Autoencoder-denoised image.

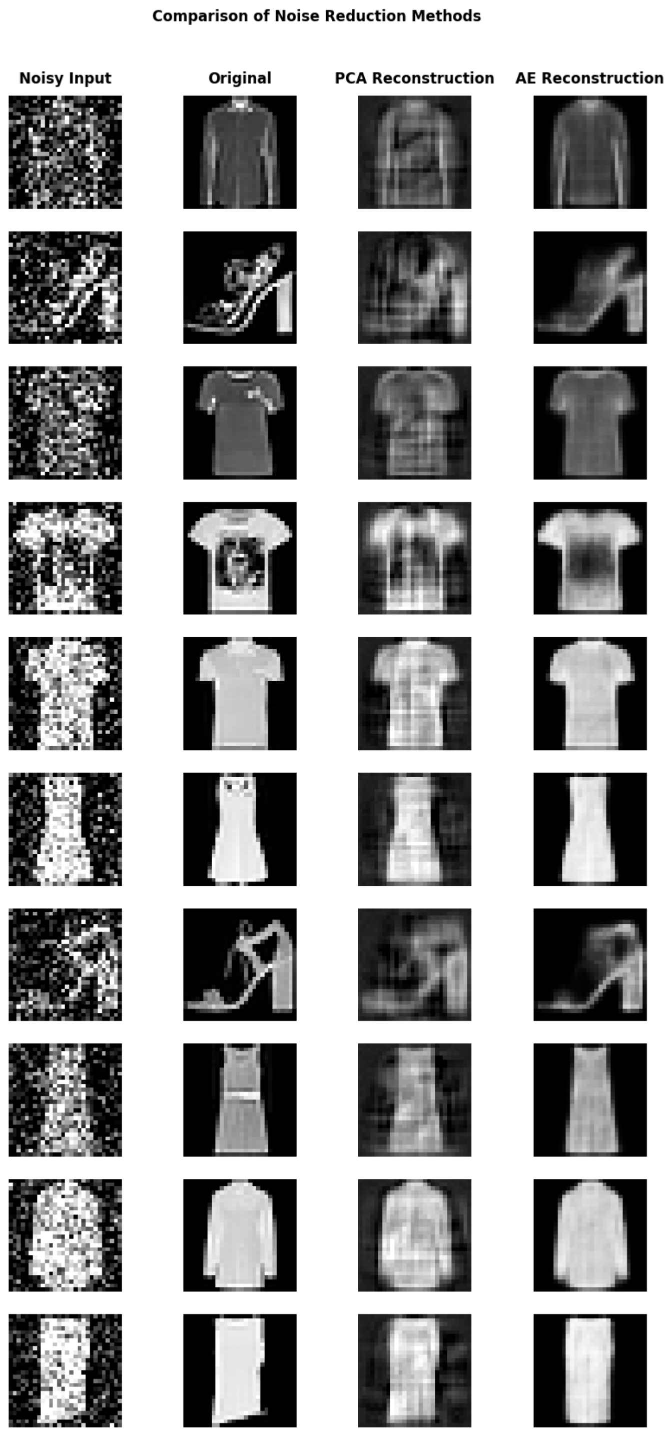

Figure 5: Comparison of noise reduction results

## 7 A Challenge: Fair Comparison

The biggest challenge in this assignment was ensuring a fair comparison. The dataset was downloaded once, and noise was added only once to ensure consistency across methods. This was critical because Scikit Learn was used for PCA, and PyTorch was used for the Autoencoder. Adding noise separately for each method could lead to inconsistencies, and loading the dataset separately for each method would cause redundancy. Thus, the dataset was downloaded and noised once, used consistently throughout, ensuring a fair comparison.

## 8 Conclusion

Based on the results in Section 6, both methods effectively reduced noise, but the Autoencoder outperformed PCA. This is likely due to the Autoencoder's ability to capture nonlinear relationships, unlike PCA, which is limited to linear relationships.