Exploring Dimensionality Reduction for Enhancing Classification Performance based on PCA and LDA

Zezheng Zhang

October 20, 2024

Contents

1 Introduction

This report explores the impact of PCA and LDA on classification accuracy as well as training time based on Iris and MNIST dataset by linear classifier and nearest neighbor classifier. Plus analysis of related results and questions.

2 Experimental Methodology

2.1 Dataset Selection and Classifiers

The following public datasets were used in this experiment:

- Iris Dataset: Contains 4 features and 3 classes. (features: 4)
- MNIST Dataset: 28x28 grayscale images, 10 classes. (features: 784)

The following classifier were used in this experiment:

- Linear Classifier
- Nearest neighbor Classifier

2.2 Data Preprocessing

Data was split into training and testing sets using a 70:30 ratio.

2.3 Dimensionality Reduction Methods

$m_{dimension}$: PCA and LDA reduced to the m dimension.
In PCA, the range of $m_{dimension}$ is $[0, F]$, where $F$is the number of features, so we try to find the best $m_{dimension}$ for PCA by using loops.
In LDA, the range of $m_dimension$ is no more than $max[F, C-1]$, where C is the number of classes. So we also try to find the best $m_{dimension}$ for LDA by using loops.

2.4 Classifier Selection and Parameter Settings

In Iris Dataset, We used the linear classifier and k-Nearest Neighbors classifier and Mahalanobis Distance classifier for classification.
In Mnist Dataset, we used the linear classifier and k-Nearest Neighbors classifier because Mahalanobis Distance classifier required inverse matrix, and often lack rank when faced with high dimensional and large number of datasets.
The k-NN classifier was based on Euclidean distance

3 Experiment Results

Due to the simplicity of IRIS dataset the dimensionality that PCA and IDA can reduce is limited, we tested the Mahalanthin minimum distance classifier on the IRIS dataset, and our main experimental object was the PCA and IDA dimensionality reduction experiment based on the MNIST dataset.

3.1 Iris Dataset

Table 1: Iris PCA

m_dim	Accuracy_linear	Accuracy_knn	Eu_recon loss	Ma_recon loss
4	0.9333	0.9333	6.62E-31	1.27E-15
3	0.9333	0.9111	0.020981833	0.781020253
2	0.9111	0.8667	0.191651728	1.231132771
1	0.9111	0.8	1.171959196	1.582676669

From Table1 , when PCA from high dimension to low dimension, Euclidean distance reconstruction loss and Mahalanobis reconstruction loss increase, and Mahalanobis loss increases significantly. Both linear classifier and nearest neighbor classifier are less accurate.

Table 2: Iris IDA

m_dim	Accuracy_linear	Accuracy_knn
3	0.9778	0.9111
2	0.9778	0.8667

From Table1 and Table2 , in terms of Iris dataset, IDA is better than PCA. IDA of reducing to 3 dimension gets the best performance on both linear classifier and nearest neighbor classifier.

3.2 MNIST Dataset

3.2.1 Image display after various dimension PCA

From Figure1,2, we can observe the image modification by PCA when dimension is reduced.
Obviously, the effect of pca dimensionality reduction on the image is blurring

In the process of dimensionality decline, the loss of Euclidean distance reconstruction presents an exponential increase as in Figure 16 from appendix

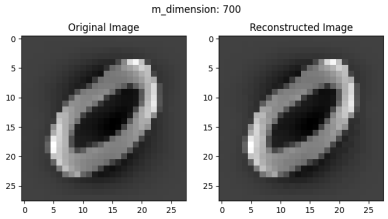

Figure 1: PCA: $m_{dimension}$=700

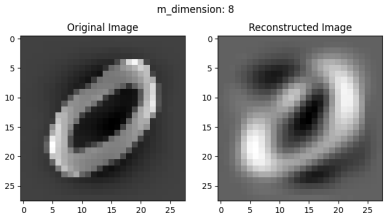

Figure 2: PCA: $m_{dimension}$=8

3.2.2 Classification Performance under Different Dimensions

1. PCA
Due to 784-dimensional features in MNIST , we first performed PCA dimensionality reduction with 14 features at intervals and calculated reconstruction losses, linear classifier accuracy, nearest neighbor classifier accuracy, and training test time from 784 dimension to 1 dimension so that we can figure out in which interval the most accurate pca dimension reduction will occur. The original accuracy is from 784 dimension of PCA.

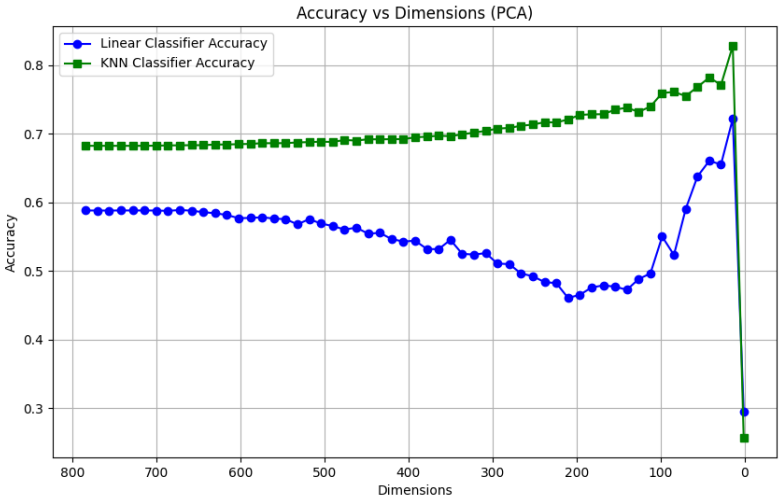

Figure 3: MNIST PCA 784-1

From Figure 17 , Know that the original classification accuracy was 0.5881 and 0.6823 respectively, we can observe that the accuracy did not decline linearly during the dimensionality decline as it did in the Iris dataset. For the nearest neighbor classifier, the accuracy rises slowly and peaks within the 50-1 dimension. For linear classifiers, accuracy slowly declines and begins to improve at 200 dimensions, reaching a maximum within 50-1 dimensions.

In terms of training and testing time from Figure 19 in appendix, there is a linear downward trend on the whole, and the lowest point is also achieved in the 50-1 dimension

Because of the above experimental results, we need to conduct a more refined PCA dimensionality reduction for dimensions 50-1, and the difference between dimensions is 1 this time

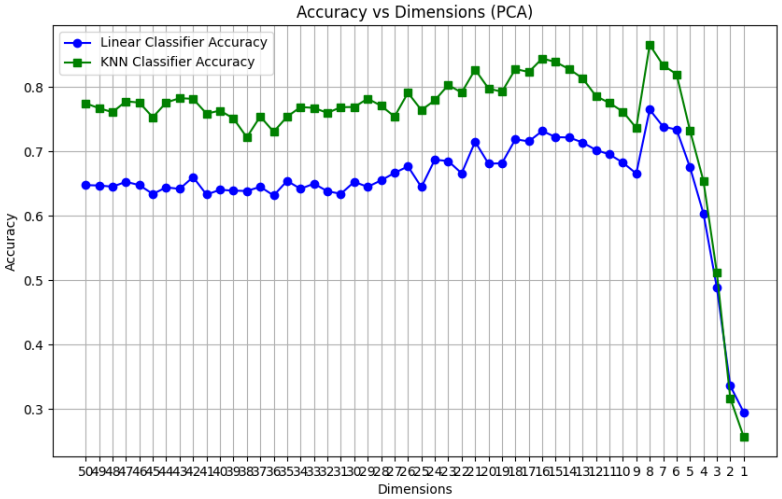

Figure 4: MNIST PCA 50-1

As can be seen from Figure18, when the decreasing dimension is 8, the accuracy of both linear classifier and nearest neighbor classifier reaches the maximum value of 784-1 dimensions. Therefore, PCA dimensionality reduction when the decreasing dimension is 8 is selected as the best PCA dimensionality reduction parameter.
And detail accuracy is shown as below:

Table 3: MNIST PCA 8 Dim Accuracy

m_dim	Accuracy_linear	Accuracy_knn
8	0.765	0.8657

Comparing to Original classification accuracy (0.5881 — 0.6823) , The accuracy increase rate of linear classifier is 17.69%, and that of nearest neighbor classifier is 18.34%. From the point of view of accuracy value, the nearest neighbor classifier is more accurate than the linear classifier in most dimensions, and is more suitable for multi-feature large data sets. The training and test time for PCA=8 is 6.5072 seconds.

2. LDA

From the LDA principle, we can find that the dimensionality reduction dimension does not exceed the maximum of the feature number or the class number -1. In Figure 5 From the dimension 9-2 process that the nearest

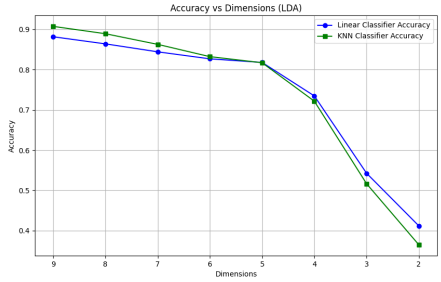

Figure 5: MNIST LDA Accuracy

Table 4: MNIST LDA 9 Dim Acc

m_d	Acc_linear	Acc_knn
9	0.8817	0.9075

neighbor classifier performs better than the linear classifier in the higher dimensions, while the opposite in the lower dimensions. And compared with PCA, LDA accuracy in 9 dimension is a little better than PCA.

Plus, training and test time for 9 dimension is 17.427 seconds, almost three times comparing to PCA.

4 Conclusion

4.1 Key Findings

When dealing with small data sets such as Iris, IDA dimensionality reduction is better than PCA dimensionality reduction. In the face of large data sets, MNIST was the main factor in the experiment, and it was found that the best dimensionality reduction of PCA was 8, and the best dimensionality reduction of IDA was 9. After the dimensionality reduction of PCA and IDA, the accuracy of IDA was better than that of PCA, but the time cost of IDA was higher, about three times that of PCA. Nearest neighbor classifiers are better suited than linear classifiers.

Because the dimensionality reduction dimension of IDA is classified by the number of categories of the dataset, the dimensionality reduction dimension of a dataset with a large amount of data but few categories, such as MNIST, is greatly limited.

4.2 Analysis

1. Which reducing dimension method is preferred?
Because the dimensionality reduction of IDA is classified by the number of categories of the dataset, the dimensionality reduction dimension of a dataset with a large amount of data but few categories, such as MNIST, is greatly limited. Though accuracy suggests the opposite, to large datasets with few classes, PCA should be preferred [1].

2. Why IDA is a little better than PCA but the training time is huge?
IDA's core goal is to maximize inter-class differences while minimizing intra-class differences by linearly combining features, but when data sets are very large (such as with millions of data points and high-dimensional features), calculating the inverses of the LDA's in-class and inter-class covariance matrices becomes more time-consuming.

3. Why accuracy of LDA shrinks so fast in dimension 10-1?
The dimensionality reduction effect of LDA is highly dependent on the linear separability between classes. If the boundaries between categories are not linearly separable, or if the categories are highly overlapping, LDA may not do a good job. This results in the interclass distribution remaining cluttered after dimensionality reduction, reducing the performance of the classifier or subsequent analysis [3].

4. Class question1: If higher accuracy can be achieved with less discriminative information, why is the criterion of dimensionality reduction (DR) set to extract the most discriminative information?
The aim of dimensionality reduction, especially in methods like LDA, is to focus on maximizing the discriminative power of the data with minimal loss of important information. If higher accuracy can be achieved with "less" discriminative information, this could imply that the method is overfitting to unnecessary details or noise in the data [2].

5. Class question2: If more discriminative information leads to higher accuracy, why do we need DR instead of directly using raw data?
Using raw data might contain too much information or irrelevant features that don't contribute to the model's ability to discriminate between classes. Plus, raw data may contain too much information or information in detail, which may lead to higher accuracy, but lower generalization and lower robustness of the model [4]. In this case, we should reduce dimension carefully with the consideration of the trade-off between robustness and ultility [5].

5 References

References

[1] X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

[2] X.D. Jiang, B. Mandal and A. Kot, "Eigenfeature Regularization and Extraction in Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 383-394, March 2008.

[3] X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

[4] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*.

[5] Pang, T., Lin, M., Yang, X., Zhu, J., & Yan, S. (2022, June). Robustness and accuracy could be reconcilable by (proper) definition. In *International Conference on Machine Learning* (pp. 17258-17277). PMLR.

6 Appendix

6.1 Code Examples

Complete Code can be seen on https://github.com/DarlinZZZ/6222PCALDA

Below is a code snippet used for PCA dimensionality reduction(in batch):

```
def apply_pca(X_scaled, n_components, batch_size=800):
    pca = IncrementalPCA(n_components=n_components, batch_size=batch_size)
    X_reduced = pca.fit_transform(X_scaled)
    X_reconstructed = pca.inverse_transform(X_reduced)
    return X_reduced, X_reconstructed
```

Below is a code snippet used for LDA dimensionality reduction:

```
def apply_lda(X_train, y_train, X_test, n_components):
    lda = LDA(n_components=n_components)
    X_train_reduced = lda.fit_transform(X_train, y_train)
    X_test_reduced = lda.transform(X_test)
    return X_train_reduced, X_test_reduced
```

Below is a code snippet used for calculate reconstruction loss euclidean:

```
def calculate_reconstruction_loss_euclidean(X_original, X_reconstructed):
    loss = np.mean(np.sum((X_original - X_reconstructed) ** 2, axis=1))
    return loss
```

Below is a code snippet used for calculate reconstruction loss mahalanobis:

```
def calculate_reconstruction_loss_mahalanobis(X_original, X_reconstructed):
    try:
        cov_matrix = np.cov(X_original, rowvar=False)  #
        if det(cov_matrix) == 0:  #
            print("")
            return None
        cov_inv = inv(cov_matrix)  #
        loss = np.mean([mahalanobis(x_orig, x_rec, cov_inv)
        for x_orig, x_rec in zip(X_original, X_reconstructed)])
        return loss
    except LinAlgError:
        print("")
        return None
```

Below is a code snippet used for linear classifier:

```
def linear_classifier(X_train, y_train, X_test, y_test):
    clf = LogisticRegression(max_iter=1000)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

Below is a code snippet used for Nearest Neighbors classifier:

```
def nearest_neighbor_classifier
(X_train, y_train, X_test, y_test, n_neighbors=5):
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

6.2 Experiment output

Complete output can be seen on https://github.com/DarlinZZZ/6222PCALDA

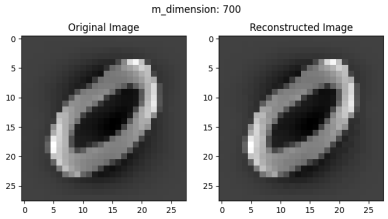

Figure 6: PCA: $m_{dimension}$=700

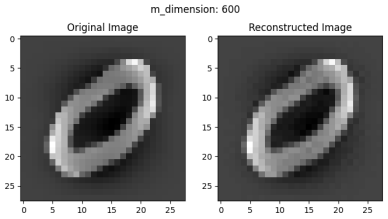

Figure 7: PCA: $m_{dimension}$=600

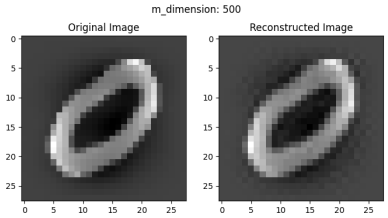

Figure 8: PCA: $m_{dimension}$=500

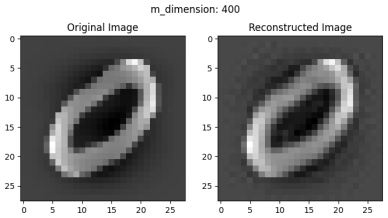

Figure 9: PCA: $m_{dimension}$=400

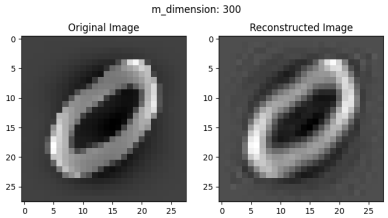

Figure 10: PCA: $m_{dimension}$=300

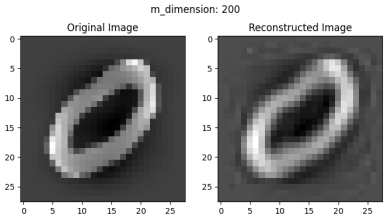

Figure 11: PCA: $m_{dimension}$=200

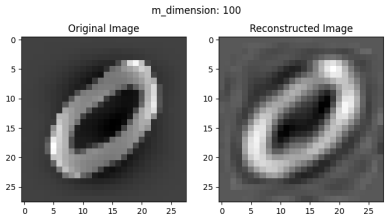

Figure 12: PCA: $m_{dimension}$=100

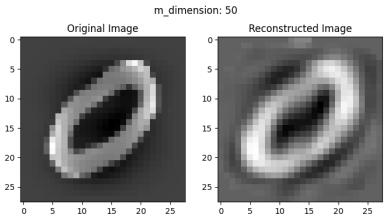

Figure 13: PCA: $m_{dimension}$=50

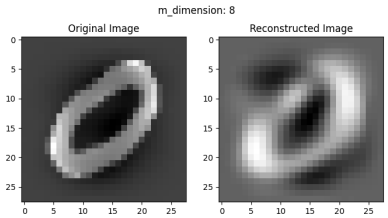

Figure 14: PCA: $m_{dimension}$=8

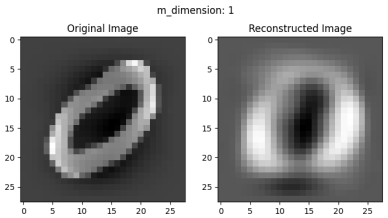

Figure 15: PCA: $m_{dimension}$=1

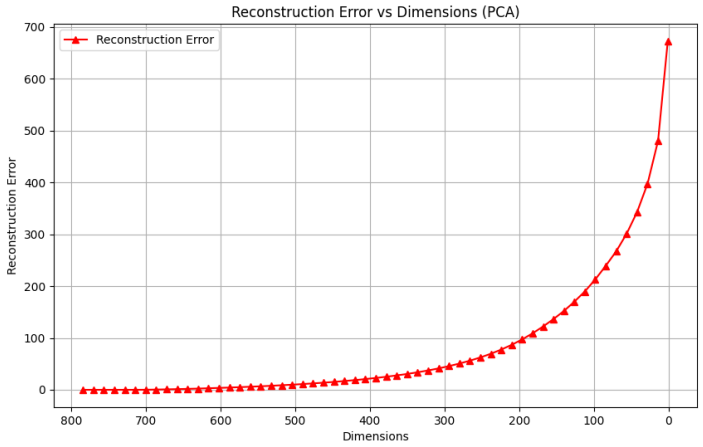

Figure 16: MNIST PCA reconstruction loss 784-1

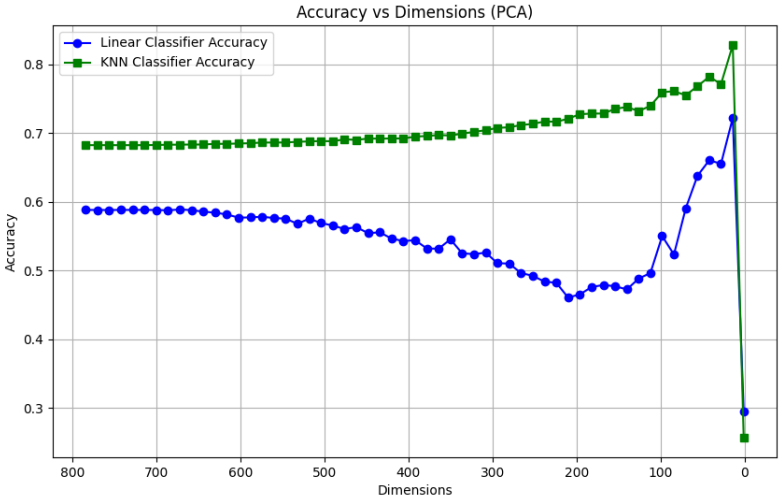

Figure 17: MNIST PCA 784-1

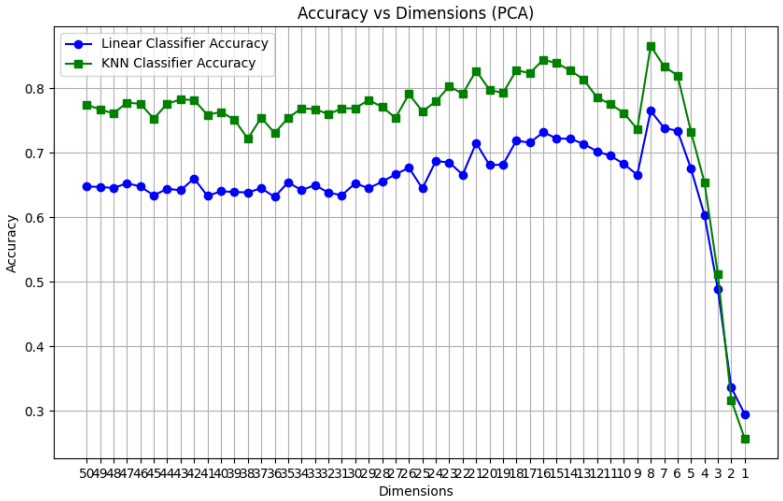

Figure 18: MNIST PCA 50-1

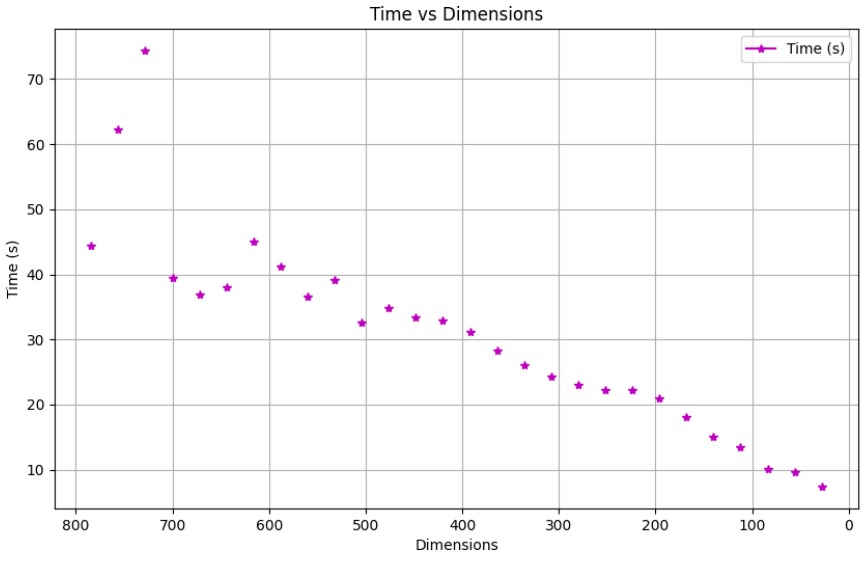

Figure 19: MNIST PCA TIME 784-1

Table 5: MNIST LDA Performance Metrics

m_dimension	Accuracy_linear	Accuracy_knn	reconstruction loss	time(s)
25	0.6447	0.7643	412.2378	7.4306
24	0.6873	0.7795	417.4255	7.7409
23	0.6851	0.8032	422.7664	6.8116
22	0.6661	0.7915	428.2353	8.0469
21	0.7152	0.8272	433.9217	7.1657
20	0.681	0.7978	439.876	6.7281
19	0.6818	0.7927	446.025	6.5828
18	0.719	0.8281	452.3644	6.3729
17	0.7157	0.8235	458.9746	6.5315
16	0.7319	0.8445	465.7169	6.5794
15	0.7223	0.8389	472.8454	11.9866
14	0.7218	0.828	480.2552	10.7813
13	0.714	0.8138	488.1336	9.3606
12	0.7023	0.786	496.1877	8.863
11	0.6955	0.7754	504.8714	8.8007
10	0.6828	0.762	514.4977	7.5789
9	0.6657	0.7366	524.5029	6.9175
8	0.765	0.8657	535.4809	6.5072
7	0.7386	0.8339	548.0672	5.9152
6	0.7341	0.82	561.8422	5.7826
5	0.6755	0.7327	577.5681	4.5662
4	0.6038	0.6545	595.679	4.627
3	0.4886	0.5122	616.4884	4.5734
2	0.3361	0.3168	643.2815	3.1215
1	0.295	0.2569	672.3401	2.8399

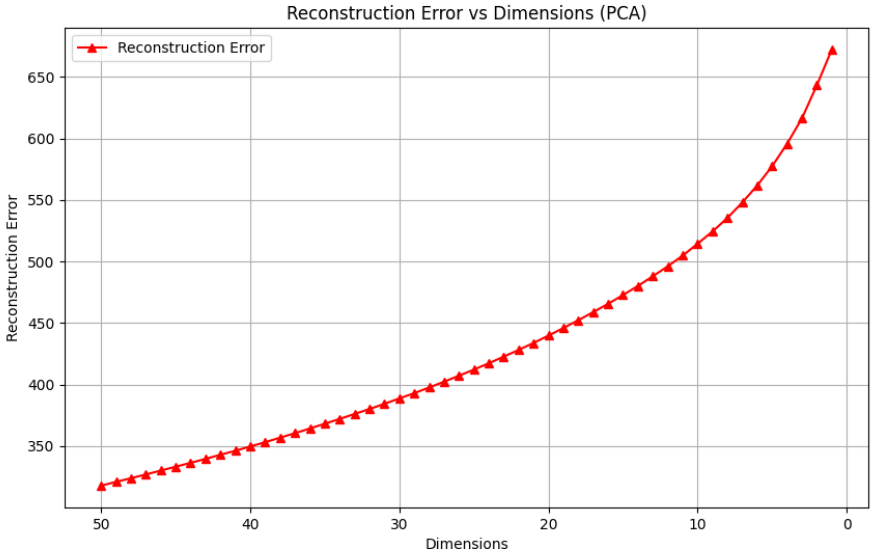

Figure 20: MNIST PCA TIME 50-1