Technical Report

Applying dimensionality reduction techniques to digit pictures.

By: Panagiotis Stenos

Contents

List of Figures

1. Aim of the Project

The aim of this project is to reduce the dimensionality of an image dataset and investigate the effectiveness of the different dimensionality-reduction models. To accomplish this, various dimensionality reduction algorithms were trained on a labeled training dataset consisting of digit images. Model performance was evaluated by measuring the test accuracy of transformed images using the K-Nearest Neighbors algorithm as the classifier. Following evaluations for different number of components, the optimal algorithm was chosen, and the advantages and disadvantages of alternative algorithms were evaluated.

2. About the Dataset

The MNIST dataset (the dataset used) is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a collection of 28x28 pixel grayscale images of handwritten digits (0 through 9). Each image is labeled with the corresponding digit it represents. MNIST stands for "Modified National Institute of Standards and Technology," referring to the original dataset created from a subset of NIST's Special Database 3 and Special Database 1.

The dataset is often employed to explore various techniques, such as dimensionality reduction, feature engineering, and different neural network architectures. Researchers use it to demonstrate and compare the effectiveness of different methodologies.

It's important to note that while MNIST has been a foundational dataset, more challenging datasets and tasks have emerged over time, encouraging the development of more sophisticated models and algorithms. However, MNIST remains a classic and valuable resource in the machine learning community. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.

3. Data Preprocessing

The data preprocessing stage was straightforward. The dimension of the feature variable is (1797, 64), signifying that the dataset consists of 1797 pictures, each with a dimensionality of 64. The figure below shows a sample of the training dataset.

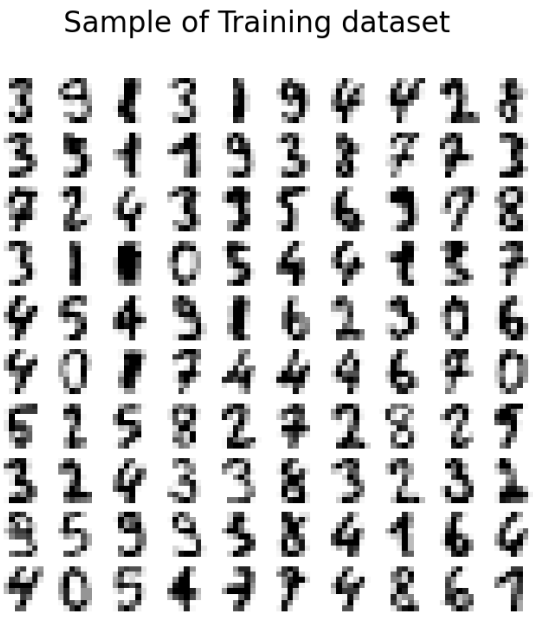

Figure 1: Sample of training dataset

4. Modelling

In this section, dimensionality reduction algorithms were applied to the dataset. The accuracy of the models was evaluated and visualized using K-Nearest Neighbors. To visually validate the performance of the algorithms, the inverse-transform of the digit pictures was obtained using the *inverse_transform()* function. Performance of models was evaluated over 2, 3, 4, 5, and 6 principal components.

a. Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning. Its primary goal is to transform high-dimensional data into a new coordinate system with a lower dimension. The new axes of this vector space are defined by considering the variance of the data in the original basis. More specifically, the algorithm finds directions in which variance is maximized and uses this direction to define the new basis for the space. This process allows for the reduction of the dimensionality of the data while retaining as much of the original variance as possible.

Before applying the algorithm, data were scaled using *StandardScaler()*, as scaling ensures that all features have the same scale, which is important for PCA because decomposition of the original matrix is based on the covariance matrix of the features. If features have different scales, those with larger scales may dominate the variance calculations, potentially leading to biased results.

b. Kernel PCA

KernelPCA is an extension that incorporates the use of kernel functions to handle non-linear relationships within the data. While traditional PCA relies on linear transformations to capture the maximum variance in the original feature space, KernelPCA enables the exploration of non-linear structures by implicitly mapping the data into a higher-dimensional space using a chosen kernel. The key difference lies in the ability of KernelPCA to capture complex, non-linear patterns that PCA may struggle with. This becomes particularly advantageous when dealing with datasets that exhibit non-linear relationships or are characterized by intricate structures. Examples include image recognition, genetics, and natural language processing. However, it's essential to note that KernelPCA may be computationally more demanding than PCA, and the choice of an appropriate kernel and its parameters becomes crucial for achieving optimal results. *StandardScaler()* was used to scale the data.

c. Multi-Dimensional Scaling

Multi-Dimensional Scaling, (MSD), is a dimensionality reduction technique employed in data analysis and visualization to represent high-dimensional data in a lower-dimensional space while preserving the pairwise distances between data points. The fundamental idea behind MDS is to capture the essential geometric relationships among data points by projecting them into a lower-dimensional space, rather than focusing on preserving variance as in PCA. The resulting lower-dimensional representation facilitates a more interpretable and visually accessible view of the underlying patterns in the dataset. *StandardScaler()* was used to scale the data. Unfortunately, sklearn MSD object, does not have a stand-alone *transform()* method like PCA making it impossible to transform the test data after fitting to the train data.

d. Isomap

Isomap is another dimensionality-reducition object from sklearn.manifold, designed to capture the intrinsic geometry of complex, nonlinear manifolds in high-dimensional data. By preserving geodesic distances through neighborhood information, Isomap is particularly useful in scenarios where the underlying structure of the data is best represented by a nonlinear manifold, such as in the analysis of sensory data, facial recognition, or any dataset with intricate nonlinear relationships. Unlike MDS,

Isomap has a stand-alone *transform()* method allowing for measuring and visualizing the accuracy of the algorithm, but like MDS, it lacks an *inverse_transform()* which means that we cannot visually assess the output of the algorithm.

e. Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is a dimensionality reduction and matrix factorization technique that is particularly useful for datasets where all the values are non-negative. The method factorizes a given non-negative data matrix into the product of two lower-dimensional matrices, where all elements in the matrices are constrained to be non-negative. Mathematically, if you have a non-negative data matrix V of size $(m \times n)$, NMF decomposes it into two matrices $W\ (m \times r)\ and\ H\ (r \times n)$, where r is the reduced dimensionality.

The goal of NMF is to find the two factor matrices W and H such that their product approximates the original matrix V:

$$V \approx WH$$

Here, W contains the basis vectors for the reduced representation, and H contains the coefficients that express the original data in terms of these basis vectors. The non-negativity constraint often leads to more interpretable parts-based representations.

Since all matrices are positive, the reconstruction of the original matrix is more intuitive, as we just add the smaller components of the matrix. Notice that because NMF has this extra constraint of positive values, it will tend to lose more information when truncating.

f. Singular Value Decomposition

Singular Value Decomposition (SVD) is a linear algebra technique that decomposes a matrix into three other matrices, providing a compact representation of the original matrix. For a given $m \times n$ matrix A, SVD factorizes it accordingly:

$$A = U\Sigma V^T$$

Where U and V are orthogonal matrices, and Σ is a diagonal non-negative matrix.

The SVD provides a way to factorize a matrix into components that capture its underlying structure. It is widely used in various applications, including dimensionality reduction, noise reduction, and solving linear systems. The singular values in Σ indicate the importance of each mode of variation in the data, and the corresponding columns in U and Vprovide the basis vectors for these modes. SVD is a fundamental tool in numerical linear algebra with applications in diverse fields.

5. Results

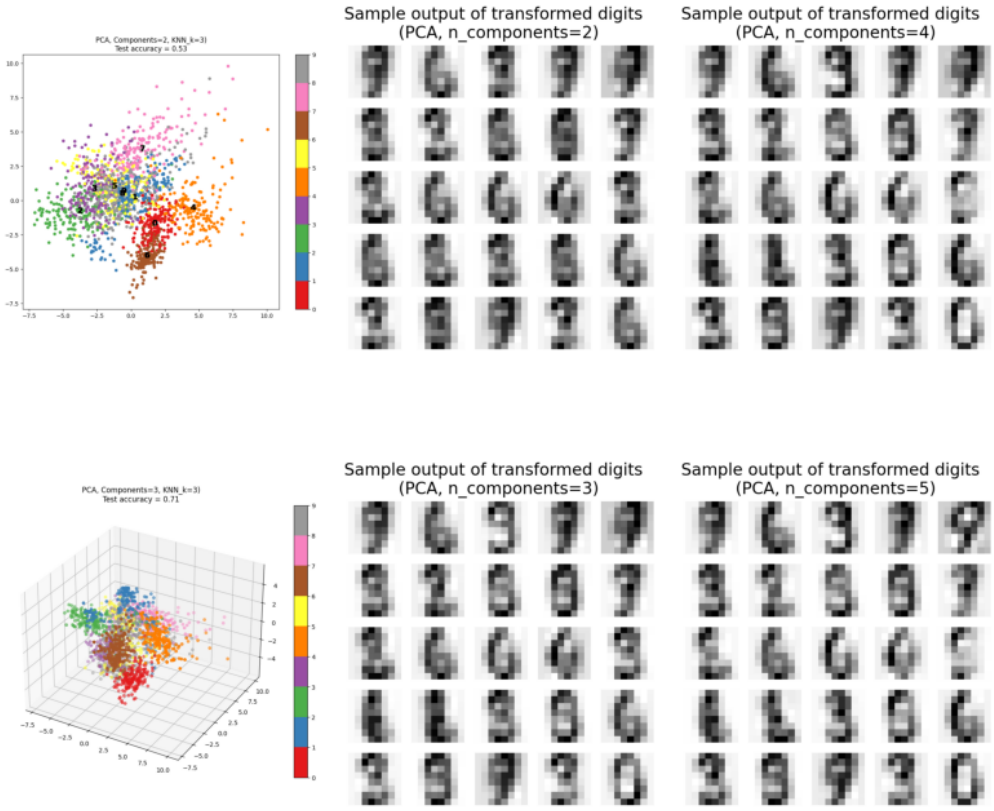

Figure 2: PCA performance

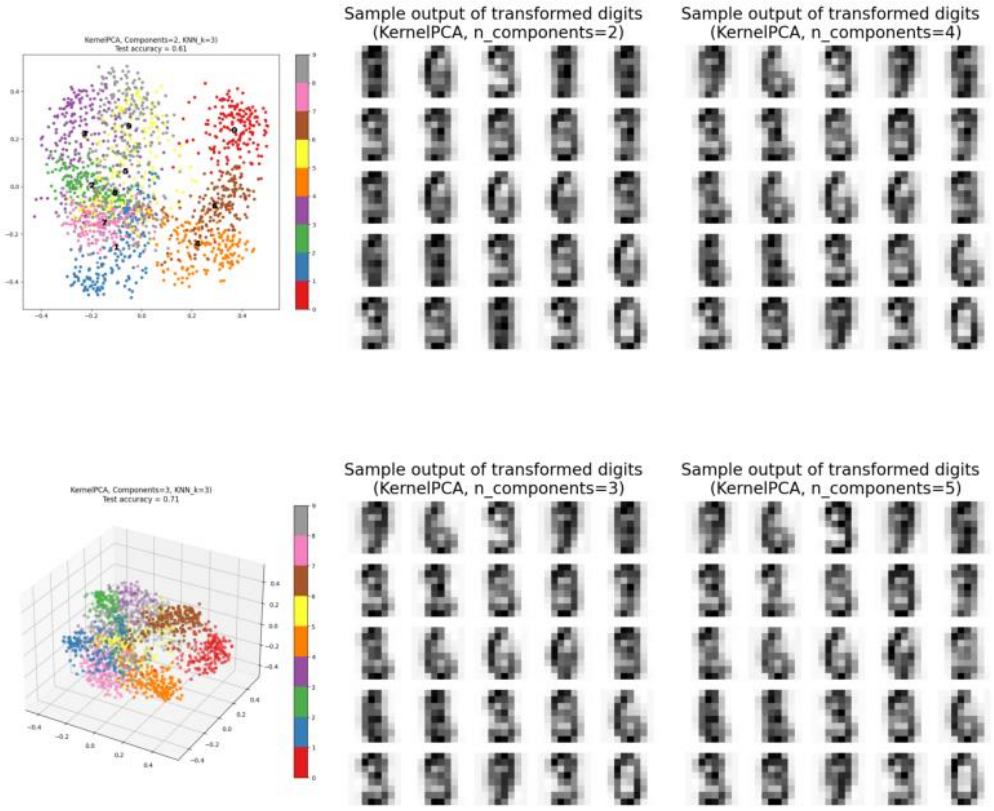

Figure 3: KernelPCA performance

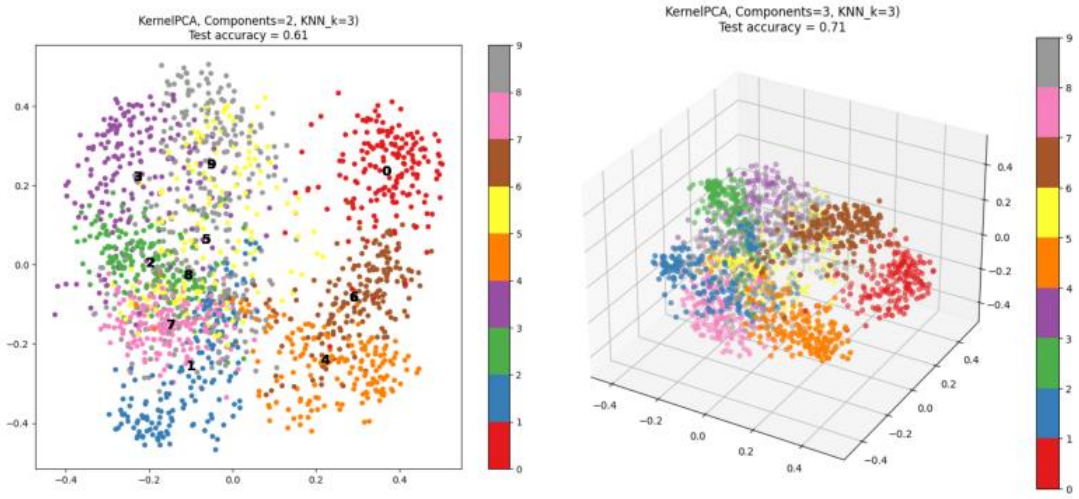

Figure 4: Isomap performance

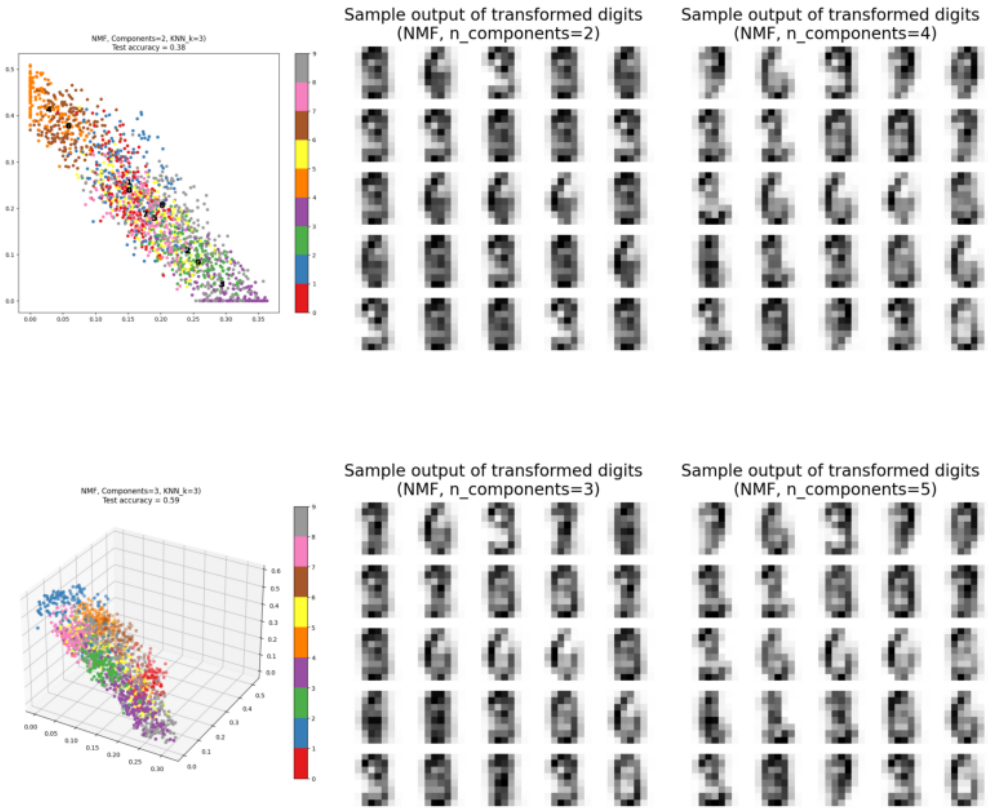

Figure 5: NMF performance

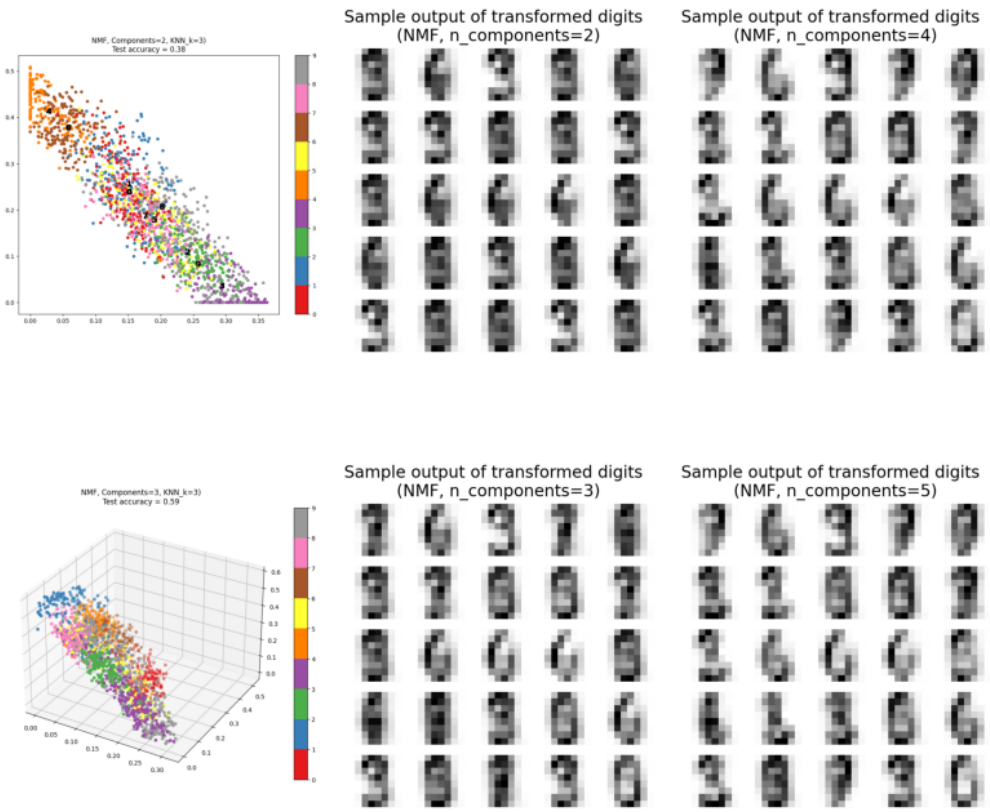

Figure 6: SVD performance

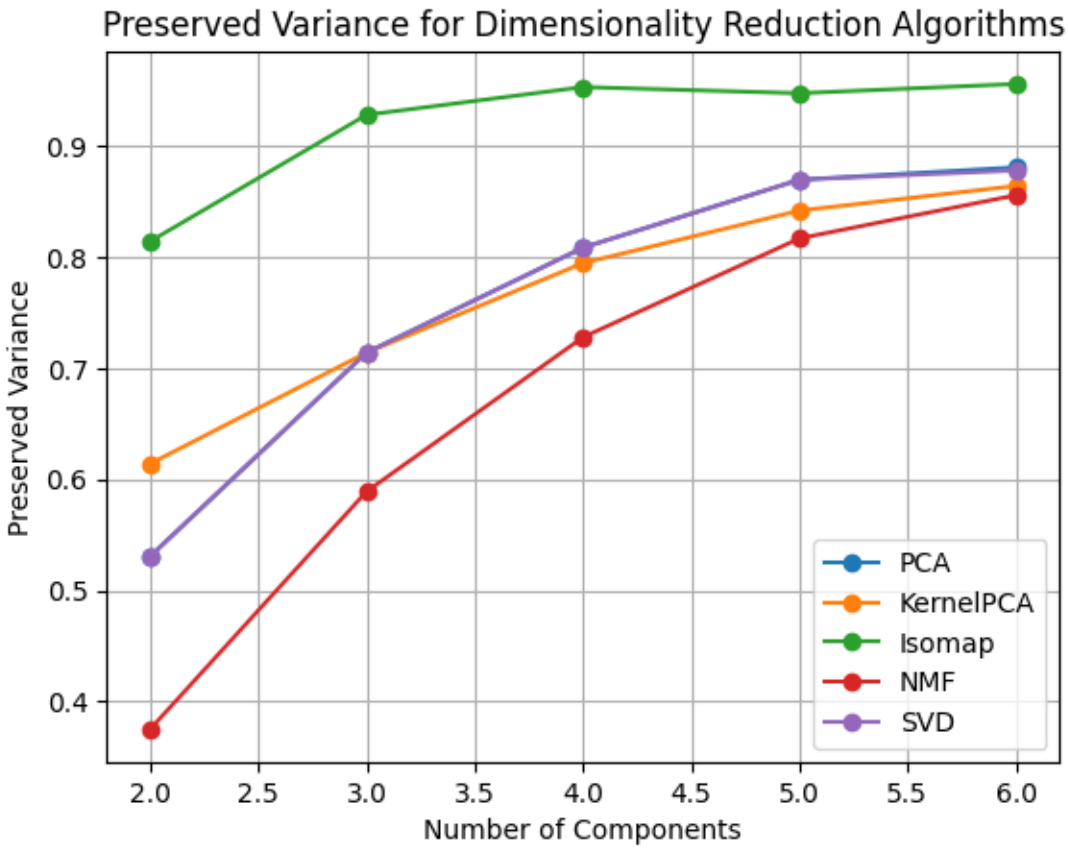

Figure 7: Preserved Variance over Number of components

6. Discussion

We observe a gradual convergence of the curves as the number of dimensions increases. It's essential to note that, given the use of test data to estimate preserved variance, the curves will never precisely converge to 1. However, they should approach the lower number close to 1 though as the number of components increase:

$$\lim_{n\_components \to 64} \approx 1$$

Based on the obtained results, the Isomap algorithm clearly stands out as the superior model. I would, therefore, use this algorithm with $n\_components = 3$, to reduce the dimensionality of the digit pictures. In the scenario where obtaining reduced digit pictures is required and considering that Isomap lacks an *inverse_transform()* method, I would utilize PCA with $n\_components = 5$. Having the ability to transform the digit pictures back to their original space is crucial for a meaningful comparison between the original digits and their transformed counterparts. Without this capability, assessing the quality of the dimensionality reduction or transformation process would be challenging. The ability to compare the original digits side by side with their transformed versions enables a visual evaluation of how well the algorithm preserves the essential features of the data during the reduction process.

The NMF algorithm performed worse than the other dimensional reduction models. In cases where interpretability and transparency in feature extraction are crucial, NMF's intuitive nature becomes especially useful. This algorithm is particularly well-suited for scenarios where the goal is to uncover meaningful patterns or representations in the data, allowing practitioners to directly interpret the contributions of different components in the factorization process. Some applications include topic modeling, face recognition, and signal processing, where understanding the composition of the latent features is essential for making informed decisions or drawing meaningful conclusions from the reduced-dimensional representation.

Lastly, similarities in the metrics of the PCA and SVM algorithms across all principal components are noticeable. This suggests a common preprocessing or feature extraction step between the two algorithms. The shared metrics likely result from applying PCA for dimensionality reduction as a preprocessing step before performing feature extraction by feeding the processed data into the SVM.