T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

NOISE REDUCTION WITH PRINCIPAL COMPONENT ANALYSIS

SENA ÖZBELEN

SUPERVISOR
DR. TÜLAY AYYILDIZ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

NOISE REDUCTION WITH PRINCIPAL COMPONENT ANALYSIS

SENA ÖZBELEN

SUPERVISOR
DR. TÜLAY AYYILDIZ

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 09/06/2024 by the following jury.

JURY

Member
(Supervisor) : Dr. Tülay AYYILDIZ

Member : Prof. Dr. Didem GÖZÜPEK

ABSTRACT

With the rise of Data Science, the methods to handle real-world datasets to extract information are improving daily. One of the biggest problems in getting the underlying pattern of the dataset is uncleaned data. Noise mostly occurs in the dataset, affecting the quality of the information extraction process. Reducing noise is an essential step for data preprocessing. In this project, noise reduction is done by a dimensionality reduction technique called Principal Component Analysis on images. For high-dimensional datasets, dimensionality reduction techniques are used to avoid "Curse of Dimensionality". These techniques are also able to reduce noise by removing the components that have small variance, possibly noise while reducing the dimension of the dataset.

Keywords: Noise Reduction, Principal Component Analysis, Image Denoising.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
PCA	:	Principal Component Analysis
KPCA	:	Kernel Principal Component Analysis
MNIST	:	Modified National Institute of Standards and Technology
KMNIST	:	Kuzushiji-MNIST
μ	:	Lowercase mu
σ	:	Lowercase sigma
Σ	:	Uppercase sigma
$\mathbb{R}$	:	The set of real numbers
α	:	Lowercase alpha
λ	:	Lowercase lambda
ϕ	:	Lowercase phi
γ	:	Lowercase gamma

CONTENTS

LIST OF FIGURES

1. PROJECT DEFINITION

Noise reduction is an important process for the preprocessing of the dataset. To extract the underlying pattern of data, there is a need for noise-free, ready-to-use data. There are a lot of noise-reduction techniques for different types of data. In this project, PCA is used to reduce noise. Normally, PCA is used for dimensionality reduction problems but for some cases, PCA is used to remove noise as well. This can be achieved by removing the components that have small variances. It is known that each component has a value for variance (a measure of how far a set of numbers is spread out from their average value). The components that include the fundamental information have the most variance in the dataset. The rest of the components have a small amount of variance, mostly tiny details or noise. When dimensionality reduction is applied, these components that have small variances are removed. This project aims to reduce noise by using this technique on images as shown in Figure 1.1.

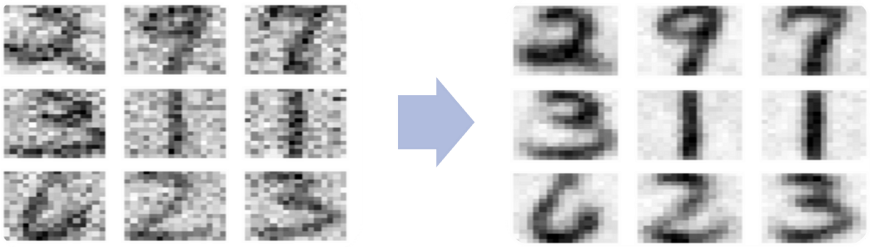

Figure 1.1: Noise Reduction in Images[1]

2. PROJECT DETAILS

2.1. Noise Reduction

Noise reduction is an important step in data preprocessing. Noise is a general term for unwanted modifications that a signal may suffer during capture, storage, transmission, processing, or conversion. Noisy data is data that is corrupted, and distorted. This dramatically affects the quality of the dataset and the information extraction process. This may cause systems to fail to interpret the dataset correctly. Therefore, there is a need for noise reduction.

2.1.1. Noise Reduction in Images

2.1.1.1. Noise Types

Image noise can result from various sources such as external factors (poor lighting or nearby electronic interference), any issues with the sensor used in cameras and scanners, conversion of analog signals into digital form, transmission, or processing.[2]

The most common noise types are ”Gaussian noise” and ”Salt and Pepper noise” as seen in Figure 2.1.[3]

- Gaussian noise is Gaussian-distributed noise. In other words, the values that the noise can take are Gaussian-distributed. It has two parameters to adjust the noise level. σ is the standard deviation and μ is the grey mean value while x is the grey value.

  $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

  Principal sources of Gaussian noise in digital images arise during acquisition e.g. sensor noise caused by poor illumination and/or high temperature, and/or transmission e.g. electronic circuit noise.

- Salt and Pepper noise is also known as impulse noise. It presents itself as sparsely occurring white and black pixels. This type of noise can be caused by analog-to-digital converter errors, bit errors in transmission, etc.

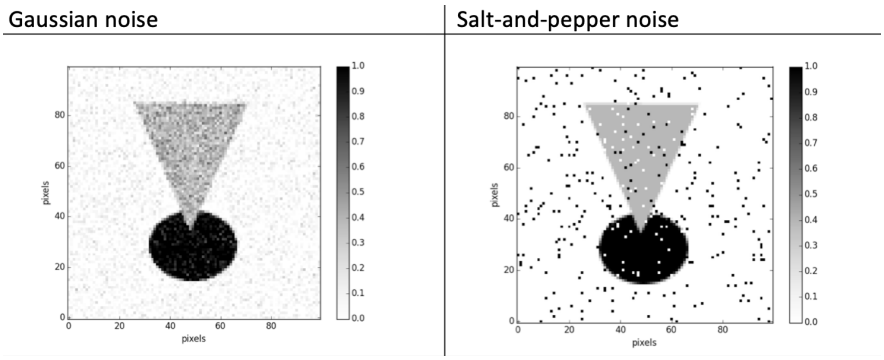

Figure 2.1: Gaussian noise vs Salt and Pepper noise [4]

2.1.1.2. Noise Reduction Techniques

There are some techniques to remove noise. With the help of image processing, filters can be used to remove noise.

- As for Gaussian noise, one method is by convolving the original image with a mask that represents a low-pass filter or smoothing operation. These are generally known as linear smoothing filters. A smoothing filter sets each pixel to the average value, or a weighted average, of itself and its nearby neighbors.
- As for Salt and Pepper noise, linear smoothing filters are unsuitable because this noise type has a value of either 0 or 255 in a greyscale image. These values directly affect the average. Instead, there is another filter type, non-linear filters. One example of a non-linear filter is the median filter which is used to remove Salt and Pepper noise. Instead of taking an average of itself and its nearby neighbors, it sorts these values and replaces the original value of the pixel with the median value of them.

2.2. Principal Component Analysis

PCA is a simple, non-parametric feature extraction method that transforms the data $X = \{x_1, x_2, \ldots, x_N\}$ from a higher-dimensional space $\mathbb{R}^M$ to a lower-dimensional space $\mathbb{R}^K$, where N represents the total number of samples or observations, and x_i represents the i^{th} sample, pattern, or observation.

Basically, its goal is to represent the dataset as a set of new orthogonal variables called principal components and to reveal hidden connections and relations by maximizing the variance on the components as shown in Figure 2.2. These principal components are uncorrelated and ordered so the first few components have most of the variation.

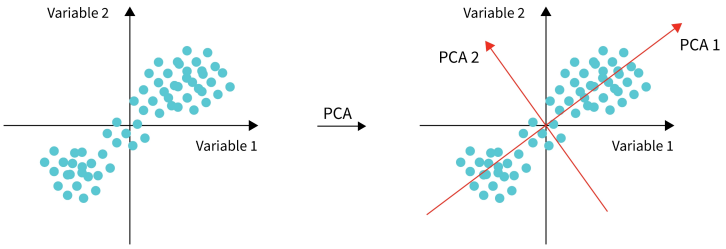

Figure 2.2: Principle Components

PCA uses a vector space transform to get these orthogonal variables and so as to reduce the dimensionality of large data sets also known as "Dimensionality Reduction". Its most common use is trying to analyze large datasets since it helps visualize the dataset. Also, it can improve the performance of the Machine Learning models by decreasing the complexity.

PCA asks if there is another basis, a linear combination of the original basis, that best re-expresses our data set shown in Figure 2.3.

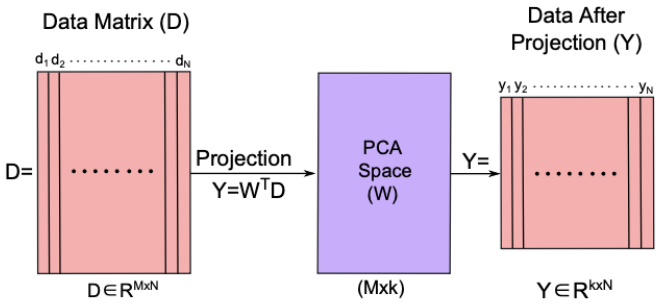

Figure 2.3: PCA in general

2.2.1. Linear PCA

It is based on linear transformation. It basically finds a line that best fits the data while being orthogonal to the other lines. These lines are the principal components of the dataset. The first principal component is the line formed as a linear combination of the original variables that explains the most variance. The second principal component explains the most variance in what is left once the effect of the first component is removed.

In the mathematical background [5], it finds $\alpha_k' x$ that maximizes $\text{Var}(\alpha_k' x) =$

$\alpha_k' \Sigma \alpha_k$ where α is the principal components, x is data and Σ is the covariance matrix of the dataset. It chooses normalization constraint, namely $\alpha_k' \alpha_k = 1$ (unit length vector).

To maximize $\alpha_k' \Sigma \alpha_k$ subject to $\alpha_k' \alpha_k = 1$ we use the technique of Lagrange multipliers. We maximize the function

$$\alpha_k' \Sigma \alpha_k - \lambda(\alpha_k' \alpha_k - 1)$$

w.r.t. to α_k by differentiating w.r.t. to α_k.

$$\frac{d}{d\alpha_k} (\alpha_k' \Sigma \alpha_k - \lambda_k(\alpha_k' \alpha_k - 1)) = 0$$

$$\Sigma \alpha_k - \lambda_k \alpha_k = 0$$

$$\Sigma \alpha_k = \lambda_k \alpha_k$$

α_k is an eigenvector of Σ and λ_k is the associated eigenvalue.

λ_1 the largest eigenvector of Σ and α_1 the corresponding eigenvector then the solution to

$$\Sigma \alpha_1 = \lambda_1 \alpha_1$$

is the 1^{st} principal component of x.

For the second component, an uncorrelation constraint is added since the components should be uncorrelated.

We can write a Lagrangian to maximize α_2

$$\alpha_2' \Sigma \alpha_2 - \lambda_2(\alpha_2' \alpha_2 - 1) - \phi \alpha_2' \alpha_1$$

Then, we are left with

$$\Sigma \alpha_2 - \lambda_2 \alpha_2 = 0$$

is the same equation as for the first principal component.

We can use the covariance matrix of the dataset to get the principal components. To do that, we need to get eigenvalues and eigenvectors of the covariance matrix by using eigen decomposition, and linear PCA is built on this logic.

2.2.1.1. Linear PCA Steps

- Data Normalization: This step ensures that each feature has the same level of contribution. This is required for absolute measurement where the actual values and differences are used. Data is normalized by applying the mean centering.
- Covariance Matrix Computation: The principle components are created with eigenvalues and eigenvectors of the covariance matrix. The covariance matrix is symmetric, and each element (i, j) corresponds to the covariance between variables i and j as shown in Figure 2.4.

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Figure 2.4: Covariance Matrix Formula

- Eigenvalues and Eigenvectors: For this part, eigenvector decomposition or singular value decomposition as shown in Figure 2.5 is used to extract the eigenvalues and eigenvectors of the covariance matrix.

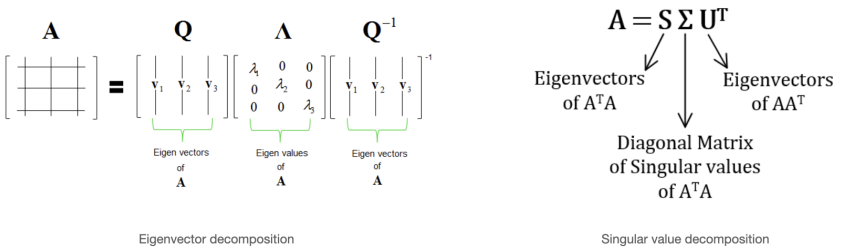

Figure 2.5: Extracting Eigenvalues and Eigenvectors

- Selection of Principal Components: In this step, what we do is, choose whether

to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors since the number of principal components is equal to the number of original variables. After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. By considering the total percentage of variance (Proportion of Variance), the number of principal components is chosen. The percentage over 90 is sufficient for dimensionality reduction cases.

- Projecting Data onto the Selected Principal Components: This step involves re-orienting the original data onto a new subspace defined by the principal components. This reorientation is done by multiplying the original data by the previously computed eigenvectors.

- Reconstruction in the Original Space: The projected data is represented by the principal components. In the case of images, to display the image, we need to reconstruct data and get the original number of features. To do that, the projected data is multiplied by the transpose of the principal components. Lastly, the mean image is added before displaying images.

2.2.2. Kernel PCA

While traditional PCA is highly effective for linear data transformations, it may not capture the underlying structure of complex, nonlinear datasets. KPCA relies on the intuition that many data sets that are not linearly separable in their current dimension can be linearly separable by projecting them into a higher-dimension space as shown in Figure 2.6. Shortly, KPCA takes our data set, maps it into some higher dimension, and then performs PCA in that new dimensional space. KPCA can effectively capture the nonlinear patterns in image data.

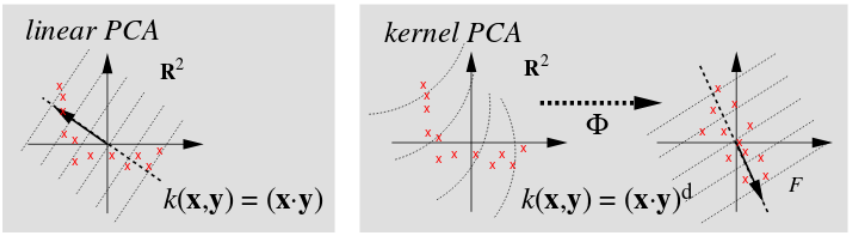

Figure 2.6: Linear PCA vs Kernel PCA

In the mathematical background [6], assume $\phi(x^{(n)})$ has zero mean. Then

consider the covariance matrix

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N} x^{(n)}(x^{(n)})^T.$$

Replacing the outer products with feature transforms

$$x^{(n)} \rightarrow \phi(x^{(n)}),$$

for some nonlinear transformation ϕ.

If this can be done, then the covariance will become

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N} \phi(x^{(n)})\phi(x^{(n)})^T.$$

However, we don't know the nonlinear transformation function ϕ. Also, it is not efficient to find and compute this function. Therefore, we use another method that allows us to work in a higher-dimension space without explicitly computing the transformation.

Kernel methods, without ever computing the coordinates of the data in that space, simply compute the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "kernel trick". The kernel trick is called this way because the kernel function enables us to get to the eigenvalues and eigenvector without actually calculating $\phi(x)$. [7]

$$k(x^{(n)}, x^{(m)}) = \phi(x^{(n)})^T\phi(x^{(m)}).$$

This creates an N-by-N kernel which is also called a "Kernel Matrix" (or Gramian Matrix). We can calculate the eigenvalues and eigenvectors of the kernel matrix and these are the new principal components of the $\phi(x)$-space where we mapped our original variables into.

2.2.2.1. Kernel Functions

- Polynomial Kernel : $k(x, y) = (x^T y + c)^d$

  Constant term: This is a regularization parameter that controls the trade-off. It is used to shift the kernel function.

  Degree: The degree of the polynomial controls the complexity of the transformation. Higher degrees can capture more complex, non-linear relationships.

- Sigmoid Kernel : $k(x, y) = \tanh(\gamma x^T y + c)$
  Gamma: This parameter is a scalar quantity. It influences how sensitive the kernel is to the input data.
  Constant term: This is a regularization parameter that controls the trade-off. It is used to shift the kernel function.

- Radial Basis Function (RBF) Kernel : $k(x, y) = \exp(-\gamma \|x - y\|^2)$
  Gamma: This parameter controls the width of the kernel (or how much influence each data point has). When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data.

2.2.2.2. Kernel PCA Steps

- Data Normalization (the same as Linear PCA)
- Kernel Matrix Computation: Instead of using a covariance matrix, we compute the kernel matrix by using kernel functions and this will allow us to work as if we are in high dimensional space.
- Eigenvalues and Eigenvectors (the same as Linear PCA)
- Selection of Principal Components (the same as Linear PCA)
- Projecting Data onto the Selected Principal Components (the same as Linear PCA)
- Reconstruction in the Kernel Space: The projected data is represented by the principal components. In the case of images, to display the image, we need to reconstruct data and get the original number of features. To do that, the projected data is multiplied by the transpose of the principal components. However, in this case, this reconstruction results in kernel space (high-dimensional space) since the kernel matrix is used as the data matrix.
- Inverse Transform Data Back to Original Space: We need to transform data back to the original space to display images. To do that, we multiply reconstructed data by original data. This way is the basic and simple inverse transformation. This might fail in some complex cases. Lastly, the mean image is added before displaying images.

2.3. The Application

The application part consists of Linear and Kernel PCA calculations for two datasets, MNIST in Figure 2.7 and KMNIST in Figure 2.8. Both of the datasets have 10 classes, 100 images for each. They are scaled to 20x20. MNIST has handwritten digits. KMNIST has handwritten Hiragana characters.

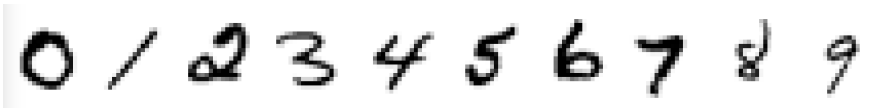

Figure 2.7: Ten class of MNIST

Figure 2.8: Ten class of KMNIST[8]

It has a PyQT5 user interface in Figure 2.9 that allows us to choose the dataset, noise level, the number of principal components, PCA type, kernel function, gamma for sigmoid and RBF functions, coef0 for sigmoid and polynomial and degree for polynomial. It loads the chosen dataset and adds some Gaussian noise which is determined by the given noise level. This level is the standard deviation of the Gaussian distribution. If this value is increased, then the noise level is increased as well. The number of principal components is important. This parameter changes the whole result. If the number of components is high, close to the number of original variables, then the result will be almost the same as the original images. This might not be valid for kernel PCA because the kernel matrix is used in PCA calculations. This means that we lose the original data.

It applies the chosen PCA steps mentioned in Linear PCA and Kernel PCA steps. Then, the program displays the results as shown in Figure 2.10.

It is already mentioned that each component has a level of variance. The program shows the cumulative variance carried by the components as shown in Figure 2.11.

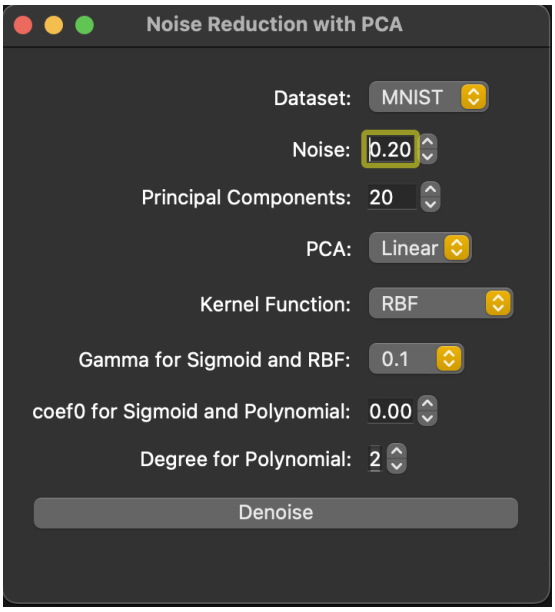

Figure 2.9: The Menu

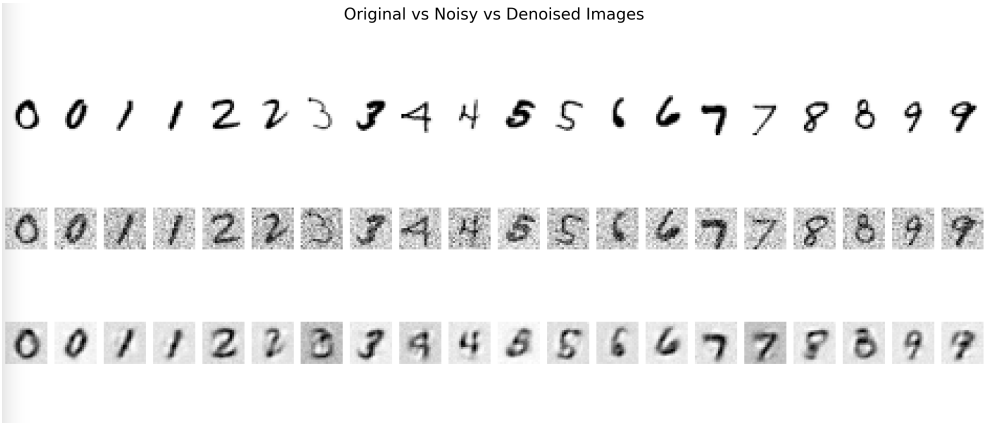

Figure 2.10: Original vs Noisy vs Denoised Images

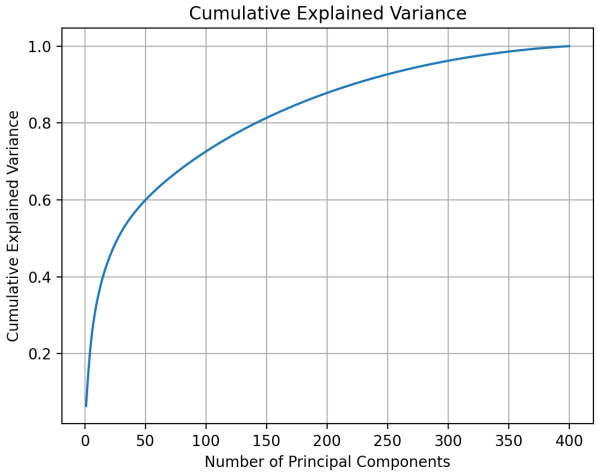

Figure 2.11: Cumulative Explained Variance

The first principal components are the components that carry the most variance. Therefore, this program shows the first 20 components if available as shown in Figure 2.12. The number of components might be lower than 20. In that case, all the components are shown in the program. The components as individuals don't give any meaning. Therefore, we can't interpret these components properly.

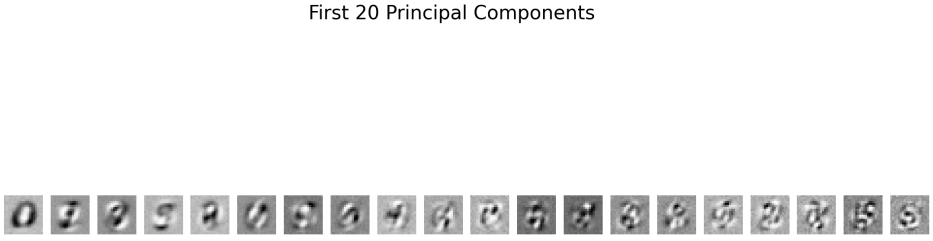

Figure 2.12: The First 20 Components

These datasets are labeled so this means that we can use the class information to interpret the first two components. We can say that if the images are in the same class, then their principal components (especially the first components) are expected to be similar. Therefore, there should be visible clusters when the first and second component values are plotted as shown in Figure 2.13 if the chosen PCA is suitable for the dataset. Kernel PCA functions and their parameters change the results a lot so it is possible that we might not see a decent clustering for each option.

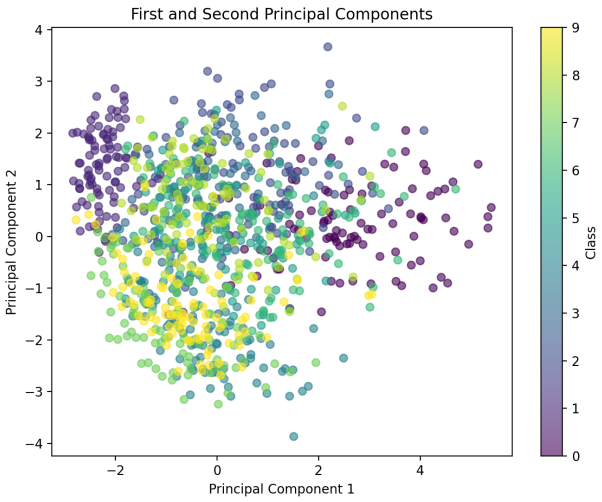

Figure 2.13: The First and Second Principal Components

There are some error criteria to evaluate the result of PCA. They are generally based on pixel-by-pixel comparison. Since PCA doesn't handle noise reduction completely, these criteria might lead to the wrong interpretation. This program has two error criteria as also shown in Figure 2.14 and 2.15.

- Root Mean Square Error (RMSE): Root Mean Square Error is a type of error measuring technique used very commonly to measure the differences between the predicted value by an estimator and the actual value. It evaluates the error magnitude. It is a perfect measure of accuracy that is used to perform the differences of forecasting errors from the different estimators for a definite variable.[9]

- Signal-to-Noise Ratio (SNR): The signal-to-noise ratio is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to noise power, often expressed in decibels. A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise.[10]

Root Mean Square Error: 0.2400 Signal-to-Noise Ratio: 2.2905 dB

Figure 2.14: The Error Results in the Program

$$\mathrm{MSE} = \frac{1}{MN}\sum_{n=0}^{M}\sum_{m=1}^{N}\left[\hat{g}(n,m) - g(n,m)\right]^2$$

$$\mathrm{RMSE}\left(\hat{\theta}\right) = \sqrt{\mathrm{MSE}\left(\hat{\theta}\right)}$$

$$SNR = 10.log_{10}\left(\frac{P_{signal}}{P_{noise}}\right)$$

Figure 2.15: RMSE and SNR formulas

To improve the results of PCA, this project utilizes some image processing techniques. Images have white backgrounds and black objects. However, the noise in the background cannot be removed completely since PCA is not capable to detect the object and remove the background noises. Firstly, the program takes the images that are the result of PCA denoising and then, it increases the contrast by multiplying with some numbers greater than 1. This will make the background of the object white because the background color is brighter than the object since its intensity value is greater. For example, we have two pixel values, 1 and 100. When they are multiplied by 2, we will get 2 and 200. This multiplication increases the value of a greater number. Similarly, when the background is multiplied by a number that is big enough, it becomes white in an ideal case. The number is chosen by considering the upper-left corner intensity value. What we get is the object itself after increasing contrast. The object part which means non-white pixels is extracted and copied to a white background by darkening the object since the contrast operation makes it brighter as well. Darkening can be done by subtracting a number from the intensity value. The result is shown in Figure 2.16. This works properly if the background is not too dark to make it white. If it is close to black, then the removal of the background cannot be done properly because this also removes the object itself. Therefore, there is a limit to increase contrast not to distort the objects.

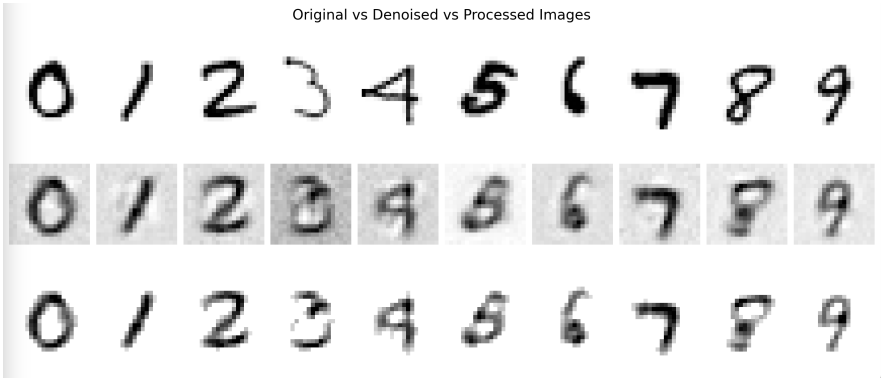

Figure 2.16: Original vs Denoised vs Processed Images

3. CONCLUSION

PCA is used to reduce noise in images. Since the main aim of PCA is not to reduce noise, we can't expect completely noise-free images as a result of this project. It basically reduces the dimension of the dataset and this also reduces noise to a certain extent. In the case of the available dataset, the background can't be totally white as the background of the original images. PCA combines the selected principal components to get the data. Noisy images have noise in the background and the results are based on the color of the noise. However, it gives a kind of smoothing effect. For the rest, it utilizes some image processing techniques. In this way, the noise is reduced. Also, for some parameters, we can observe that the result might not look the same as the original one. This distortion is normal since PCA combines the principal components to get data. It doesn't use all the components and some amount of variance is removed. As for Kernel PCA, the kernel matrix is used for PCA operations and the original dataset is not used. Since we lose the original data, there may be some distortions. Also, inverse transform might fail in some cases because it tries to catch the relation between the reconstructed data and the original data by using a basic multiplication operation. In conclusion, the right parameters for PCA and the help of image processing led us to a decent result.

BIBLIOGRAPHY

[1] Fateme Akbari. "Kernel pca: From uncovering the hidden patterns of wealth to turning noisy images into clear images." (2023), [Online]. Available: https://medium.com/the-power-of-ai/kernel-pca-from-uncovering-the-hidden-patterns-of-wealth-to-turning-noisy-images-into-clear-images-948d6a81e969.

[2] Haziqa Sajid. "What is noise in image processing?" (2023), [Online]. Available: https://www.unite.ai/what-is-noise-in-image-processing-a-primer/.

[3] Wikipedia. "Noise reduction." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Noise_reduction.

[4] Ron Dror. "Image analysis." (2015), [Online]. Available: https://web.stanford.edu/class/cs279/notes/image-analysis-notes.pdf.

[5] Frank Wood. "Principal component analysis." (2009), [Online]. Available: http://www.stat.columbia.edu/~fwood/Teaching/w4315/Fall2009/pca.pdf.

[6] Stanley Chan. "Feature analysis via pca." (2020), [Online]. Available: https://engineering.purdue.edu/ChanGroup/ECE595/files/Lecture07_pca.pdf.

[7] Wikipedia. "Kernel method." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Kernel_method.

[8] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. "Deep learning for classical japanese literature." arXiv: cs.CV/1812.01718 [cs.CV]. (Dec. 3, 2018).

[9] U. Sara, U. Akter, M., "Image quality assessment through fsim, ssim, mse and psnr," *Journal of Computer and Communications*, 2019.

[10] Wikipedia. "Signal-to-noise ratio." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Signal-to-noise_ratio.