Face Images Classification Using K Nearest Neighbors and Principal Component Analysis

Chun Li 260986765, Yujin Li 260968957

McGill University ECSE 526

***Abstract* — This project investigates the combined use of K Nearest Neighbors (KNN) and Principal Component Analysis (PCA) in face image classification. Recognizing their strengths in classification and dimensionality reduction respectively, we explore their performance under different noise scenarios. Applying this approach to human face datasets with varying noise levels, we uncover its susceptibility to noise factors like background noises, differences in viewpoints and face positioning. Additionally, we extend the study to animal face classification for species recognition, revealing limitations in the technique's adaptability to diverse datasets. Despite achieving 90.00% accuracy in a low-noise human face dataset, challenges emerge in handling poorly posed faces and varied species recognition scenarios. The technique's sensitivity to noise factors prompts the suggestion of additional preprocessing approaches to optimize performance in specific contexts.**

Keywords — eigenface, image classification, K Nearest Neighbors (KNN), Principal Component Analysis (PCA)

I. Introduction

In the realm of computer vision, the quest for robust and efficient face image classification algorithms has been an ongoing pursuit. The ability to identify and classify faces accurately is fundamental in various applications, ranging from security systems to user authentication. In this project, we embark on an investigation into the performance of a combined approach utilizing K Nearest Neighbors (KNN) and Principal Component Analysis (PCA) for face image classification. The selection of KNN and PCA stems from their respective strengths in handling classification tasks and dimensionality reduction. We would like to re-implement the approach described by [1] to extract facial features from the datasets, followed by applying the KNN algorithm to perform face classification.

However, this combination of techniques is easily affected by noises in datasets, such as differences in viewpoints and positioning of faces [2]. Therefore, we aim to address the question: How does the combination of KNN and PCA perform in the classification of face images under different noise scenarios? To investigate this, we apply the same strategy on two different human face datasets, one with low noise level, the other with high noise level, and compare how much the performance was impaired by the noise. To further extend the scope, we explore the adaptability of this technique to animal face classification for the sake of species recognition, testing its mettle on a dataset featuring non-human faces. The ultimate goal is to discern under what scenarios the combination of KNN and PCA yields optimal performance in face image classification.

II. Datasets

A. Olivetti Faces Dataset

The first dataset, obtained from the python library sklearn.datasets.fetch_olivetti_faces, consists of 400 gray scale images capturing the faces of 40 individuals. Each image has dimensions of 64x64 pixels, and the backgrounds are consistently clean. Notably, all faces are well-posed, with subjects positioned in the middle and images taken at a uniform distance, and it is therefore considered as a dataset with low noise level. Figure 1 shows 16 images selected randomly from this dataset.

[Figure 1]

B. Kaggle Avengers Faces Dataset

The second dataset, sourced from Kaggle, comprises 394 images, featuring five different individuals.

[Figure 2]

Figure 2 are 16 images selected randomly from this dataset. While the backgrounds are relatively clean, the faces are poorly posed, particularly concerning their positioning within the images. We addressed an issue where the validation and testing set contains duplicates of the training set, potentially inflating prediction accuracy. These images were consequently removed from the training set. The validation set and training set were merged for the sake of k-fold cross validation, leaving 30 to 40 training samples per individual.

C. Kaggle Animal Faces Dataset

The third dataset, extracted from Kaggle's Animal Faces collection, encompasses 16,130 images representing three distinct subjects: cat, dog, and wild. To streamline our implementation, we manually classified the "wild" category into subclasses named under different species. Due to the dataset's extensive size, we opted to use only half of the images while ensuring an equal distribution of samples across all classes. Therefore, only half of the classes, namely cat, dog, tiger and leopard, are used during classification. 16 examples are shown in Figure 3.

[figure 3]

III. METHOD

A. PCA

The implementation of Principal Component Analysis (PCA) was following the descriptions in [1]. Initially, we calculated the mean face by averaging the flattened images. The data was then centered by subtracting the mean face from each image. The covariance matrix $D^T D$ was computed to capture the relationships between features. Eigenvalues and eigenvectors were obtained using the np.linalg.eig function. The eigenvectors were normalized to have a magnitude of 1, and the results were sorted in descending order based on eigenvalues. The designated number of eigenvectors were then selected, providing a reduced-dimensional representation of the data.

Notice that in contrast to the conventional DD^T formulation, the function employed $D^T D$. The rationale for this modification lied in the dimensions of D, the matrix representing the centered data. Let the number of pixels in each image be M, and let the number of samples in the dataset be M. With D having dimensions of M×N, where the number of pixels significantly exceeded the number of samples, calculating DD^T would result in an enormous M×M covariance matrix. This operation could be computationally intensive and memory-demanding, especially when the number of pixels was substantial. By opting for $D^T D$, the resulting covariance matrix was of size N×N, which can be more manageable, particularly when the number of samples was comparatively smaller. This adjustment addressed computational efficiency considerations while still capturing the essential relationships between features in the dataset, the following was the derivation.

Given the equation of eigenvector and eigenvalue, we looked for v and λ in (1):

$$[DD^T]v = \lambda v \tag{1}$$

Multiple both sides with D^T and we got (2):

$$D^T[DD^T]v = D^T[\lambda v] \tag{2}$$

Re-arrange the brackets and we got (3):

$$[D^T D]D^T v = \lambda[D^T v] \tag{3}$$

Let the new eigenvector $v' = D^T v$ and new eigenvalue $\lambda' = \lambda$, then the eigenvector and eigenvalue that we looked for can be computed by:

$$v = Dv' \tag{4}$$

$$\lambda = \lambda' \tag{5}$$

B. KNN

To implement the K-Nearest Neighbors (KNN) classifier, we implemented the method *fit* for fitting the model with training data and labels, as well as the method predict to predict labels for test data based on the nearest neighbors. The *fit* method assigned the training data and corresponding labels to the instance variables. In the *predict* method, Euclidean distances between a test instance and all training instances were computed. The indices of the k-nearest neighbors were determined, and their labels were collected. The classifier then employed majority voting to determine the predicted label of the test instance. This KNN implementation served as a straightforward yet effective approach for classification tasks, particularly in scenarios where the decision boundary was expected to be locally smooth.

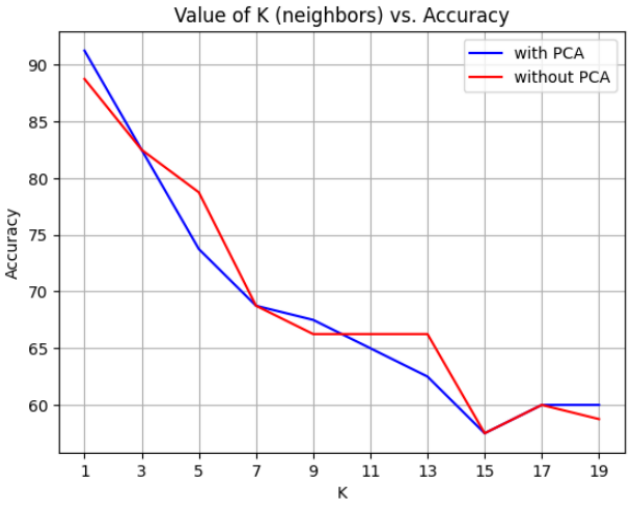
C. K-Fold Cross Validation for Improving Performance

To further enhance the model's performance, a robust approach was taken by incorporating k-fold cross-validation. Two distinct strategies were employed: one was to select the best model based on accuracy across the k-folds, and the other was to use majority voting to aggregate predictions from different folds. These strategies not only aided in fine-tuning the model but also ensured a more generalized and reliable KNN classifier, making it well-suited for a variety of classification tasks, particularly in scenarios where the decision boundary was expected to be locally smooth.

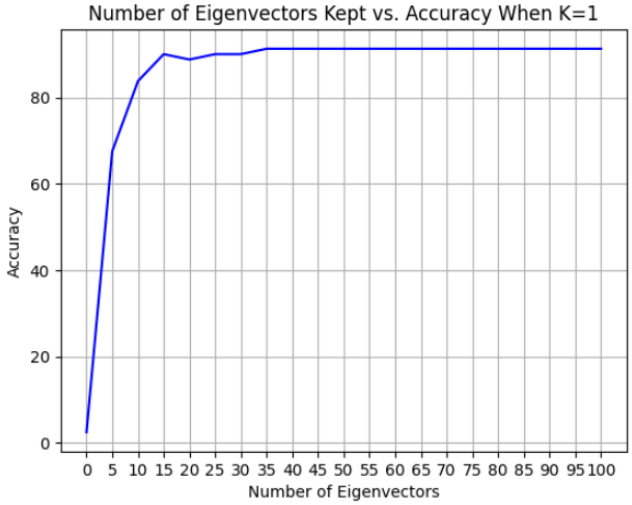
IV. Result

A. Olivetti Faces Dataset

For the Olivetti Faces dataset, applying PCA with 15 eigenvectors, followed by KNN, where K=1 achieved an accuracy of 90.00%. The decision of K=1 was based on experiments of running KNN with different values of K (number of neighbors). As shown in figure 4, the optimal performance was achieved when K was set to 1. Notice that in figure 4, the accuracy of model with PCA and the accuracy of model without PCA did not have an obvious gap.



[Figure 4]

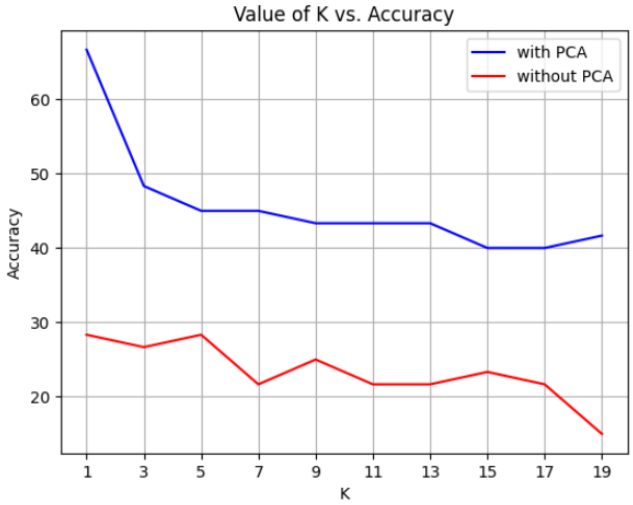


[Figure 5]

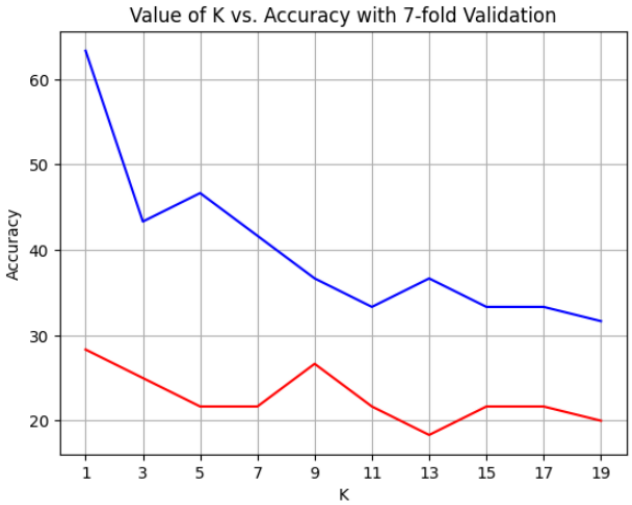
The decision to use 15 eigenvectors was based on observing that the accuracy plateau after this point, as illustrated in figure 5, indicating that the first 15 eigenvectors that were corresponding to the 15 largest eigenvalues had captured adequate features to represent the dataset.

B. Kaggle Avengers Faces Dataset

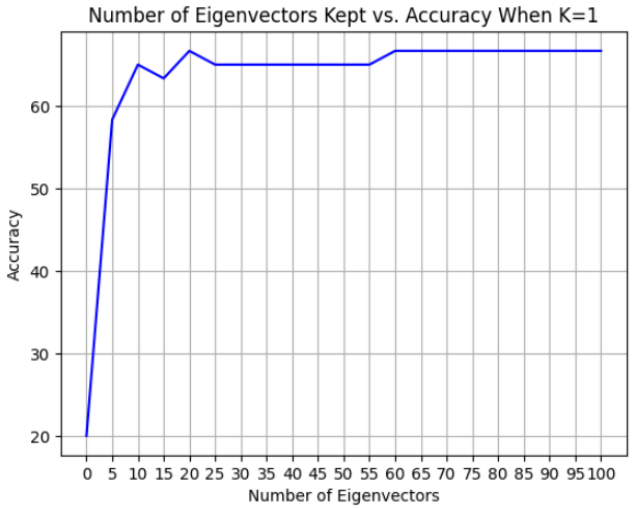
In the case of the Avengers Faces dataset, the initial configuration yielded an accuracy of 66.67% when K was set to 1 and N is set to 20. The choice is K and N were made by observing figure 6 and figure 7, following the same criteria as in the previous step.



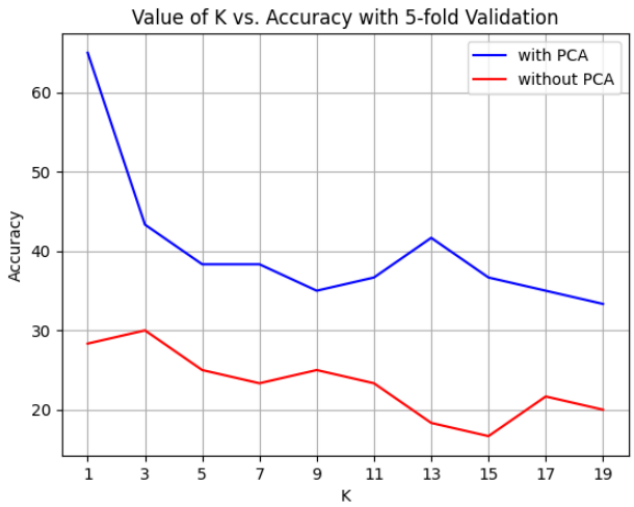
[Figure 6]



[Figure 8]



[Figure 7]



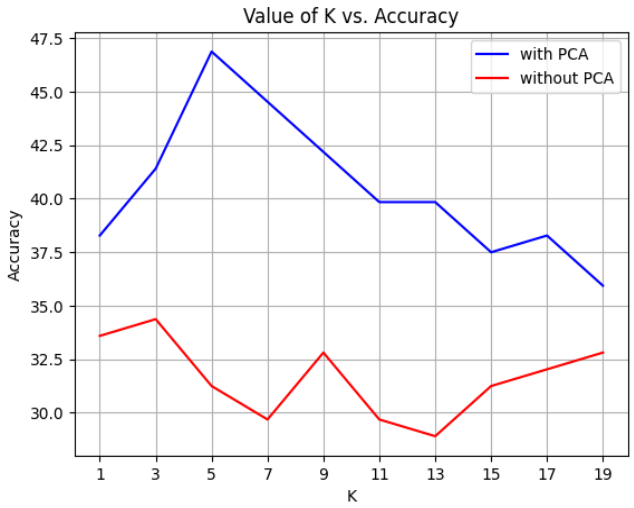
[Figure 9]

However, attempts to improve the model's performance through k-fold cross-validation, both with best model selection and majority voting, led to less favorable accuracies of 63.33% and 65%, as illustrated in Figure 8 and Figure 9, respectively. We selected 7 for the number of folds in Figure 8 and 5 for the number of folds in Figure 9 based on the best results obtained from experiments conducted with the fold values ranging from 3 to 15. It was notable that the enhancements did not yield significant improvements, and the initial configuration remains more effective for this dataset.

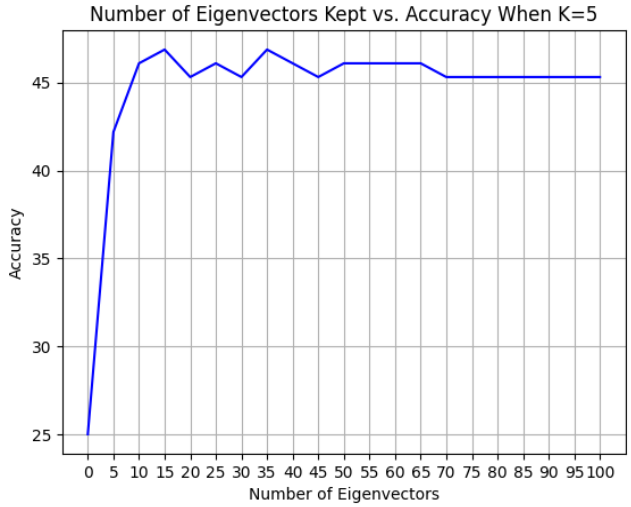
C. Kaggle Animal Faces Dataset

For the Animal Faces dataset, the model achieved an accuracy of 46.875% when k was set to 5 (obtained from Figure 10) and N was set to 15 (obtained from Figure 11).

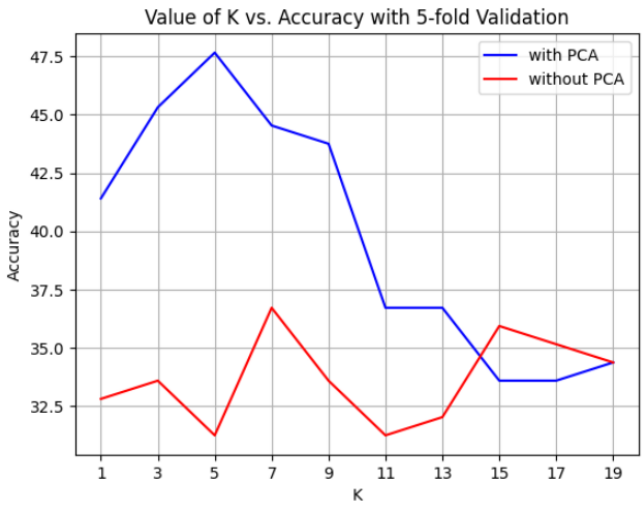
We applied k-fold cross validation with the best model and with majority voting. The results were as shown in figure 12 and figure 13, respectively. Both approaches yielded an accuracy of 47.65635%, bringing a tiny improvement to the original result. The challenges in significantly improving accuracy suggested that the inherent complexity or distribution of the dataset may limit the effectiveness of the chosen strategies.



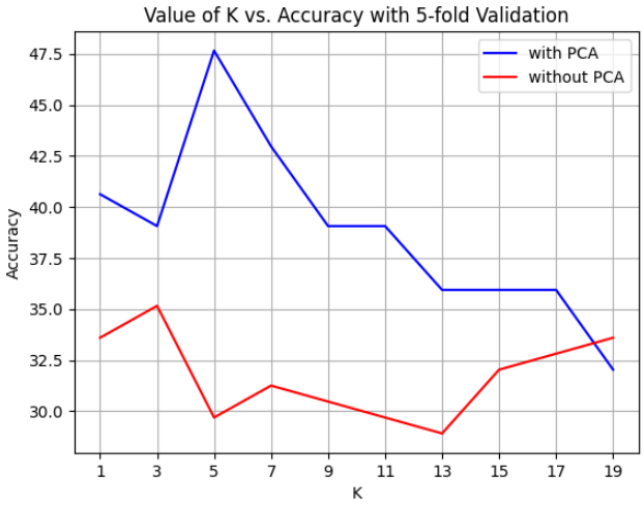
[Figure 10]



[Figure 11]



[Figure 12]



[Figure 13]

V. Discussion

A. Olivetti Faces Dataset

The success of the model on the Olivetti Faces dataset is attributed to the favorable characteristics of the data. The KNN model yielding the best performance when K is set to 1 may suggest that this dataset has a complex and non-linear decision boundary. As a result, using small K allows the model to adapt closely to the intricacy of the training samples. The absence of significant noise in the background and the well-posed nature of all faces contribute to the success of PCA in capturing meaningful features. We decide not to apply k-fold validation as the insufficient number of samples per subject might lead to underfitting.

B. Kaggle Avengers Faces Dataset

The challenges encountered with the Avengers Faces dataset are linked to the diverse positioning of faces, leading to blurry eigenfaces. This introduces difficulty in classification, and the application of k-fold validation further worsens the issue by reducing the number of training samples for each class. The limited number of samples per class, around 40, in combination with k-fold validation, potentially leads to underfitting and model instability.

C. Kaggle Animal Faces Dataset

The complexity of the Animal Faces dataset, involving multiple species, background noise, illumination, and varied face positioning, presents considerable challenges for classification. Specially, the presence of noise impacts the effectiveness of PCA. We recognize the need for additional preprocessing techniques beyond PCA for successful KNN application, such as more sophisticated feature extraction or data augmentation.

VI. Conclusion

During the experiments, we apply the combined technique PCA and KNN to three different datasets. The Olivetti face dataset, representing a human dataset with low noise level, achieves a classification accuracy of 90.00%. The Kaggle avengers face dataset gives an accuracy of 66.67% due to the noise in the positioning of faces. K-fold cross validation does not bring an improvement to this specific case. We also apply the techniques to a Kaggle animal face dataset, attempting to do species recognition, which yielded an accuracy of 46.875%. We realize that the performance of KNN combined with PCA can easily be affected by the noises like illumination, background and face positioning. This technique is therefore not ideal for species recognition, unless additional preprocesses of images, such as data augmentation or a more sophisticated feature extraction technique, are applied to improve the quality of samples.

References

[1] M. Turk and A. Pentland, "Face recognition using Eigenfaces", Proc. IEEE Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, 1991.

[2] H. Zhang and G. Chen, "The Research of Face Recognition Based on PCA and K-Nearest Neighbor," 2012 Symposium on Photonics and Optoelectronics, Shanghai, China, 2012, pp. 1-4, doi: 10.1109/SOPO.2012.6270975.