Face Recognition Using Various Feature Extraction and Classification Techniques

Parnian Taheri[a]

[a] *Sharif University of Technology, , Tehran, , ,*

Abstract

This project presents the design and implementation of an advanced face recognition system utilizing the ORL Face Database [1]. The system incorporates a multi-faceted approach, combining Principal Component Analysis (PCA) for dimensionality reduction, various feature extraction techniques including Pearson Correlation and Mutual Information, and a diverse set of classification algorithms such as k-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Parzen Window, and Random Forest. To evaluate the system's performance comprehensively, the project investigates the impact of varying the number of PCA components and the size of the training set on the accuracy of the face recognition system. This analysis provides insights into the trade-offs between computational efficiency and recognition accuracy, allowing for a better understanding of the system's robustness and scalability. The outcomes of this project contribute to the field of face recognition systems, offering a comparative study of dimensionality reduction, feature extraction, and classification techniques. The findings have potential applications in security systems, biometrics, and human-computer interaction, providing a valuable reference for future developments in facial recognition technology.

1. Introduction

Face recognition systems have become a pivotal aspect of various applications, ranging from security protocols to human-computer interaction. In the pursuit of developing more accurate and efficient face recognition technologies, this project focuses on the design and implementation of an advanced system using the ORL Face Database. By leveraging sophisticated techniques such as Principal Component Analysis (PCA) for dimensionality reduction and various feature extraction methods, including Pearson Correlation and Mutual Information, the aim is to enhance the discriminatory power of facial features.

The project extends its investigation into the classification domain by employing a diverse set of classifiers, including k-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Parzen Window, and Random Forest. Each classifier is selected for its unique attributes, and their performances are rigorously evaluated to understand their strengths and limitations in the context of face recognition.

Moreover, the dataset is partitioned equally into training and test sets (50/50%). With a fixed total size of 400 samples, this division results in training and testing sets each comprising 200 samples. This standardized approach ensures consistency in the evaluation process and enhances the reliability of the results.

One of the central focuses of this project is the exploration of the impact of varying the number of PCA components and the size of the training set on the accuracy of the face recognition system. This investigation provides valuable insights into the trade-offs between computational efficiency and recognition accuracy, offering a nuanced understanding of the system's robustness and scalability.

The outcomes of this project contribute to the existing body of knowledge in face recognition systems, offering a comprehensive exploration of dimensionality reduction, feature extraction, and classification techniques. The findings are anticipated to advance the current understanding of facial recognition technology and pave the way for more robust and adaptable systems with widespread applications in security, biometrics, and human-computer interaction.

2. ORL dataset

The ORL (Olivetti Research Laboratory) Face Database is a widely used dataset in the field of face recognition. It was created by the Olivetti Research Laboratory at the University of Cambridge and contains a collection of facial images from 40 individuals, with each person contributing 10 grayscale images, capturing variations in facial expressions, lighting conditions, and poses. The dataset was designed for research and experimentation in the development and evaluation of face recognition algorithms. The images in the ORL Face Database are relatively low resolution, typically 92 x 112 pixels. The size of images in this dataset is 48 x 48, therefore, we do not need any zero padding or other techniques to equalize the size of the images. Researchers and practitioners often choose the ORL Face Database for its simplicity, accessibility, and well-defined structure. Its standardized format and the presence of multiple images for each individual make it a valuable resource for benchmarking and comparing the performance of different face recognition methods.

3. Brief description of the used classification techniques

In this section, a brief description of each used classification techniques is presented. In the results section, the accuracy of each classifier/features combination is reported as well. Some of the used classification techniques have parameters that

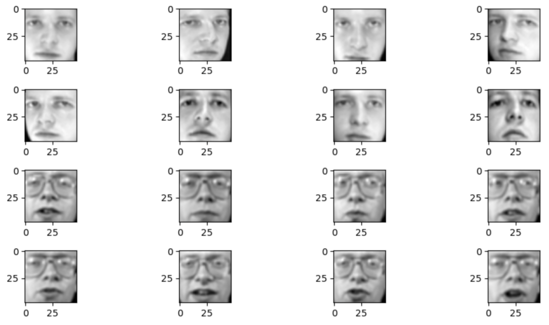


Figure 1: ORL Daataset

need to be specified (e.g. k in k-NN classifier and bandwidth in Parzen Window).

3.1. K-Nearest neighbor

The K -Nearest Neighbor (KNN) [14] is one of the simplest classification techniques; yet gives surprisingly high accuracy. In KNN, there is no training stage. All training samples must be present in the testing phase; and Euclidean distances between each training sample and the sample to be tested are calculated. The K training samples that have the smallest distances to the test sample are found and their classes are identified. To choose the best k, data is classified for different 'k's and test with the validation set and choose the one that gives the highest accuracy. In this paper, I brought both the handwritten function and python built-in function to determine the difference between their performances.

3.2. Bayes

The Bayes Classifier, rooted in Bayesian probability theory, is a powerful statistical method for classification tasks. At its core, the Bayes Classifier assigns a class label to an input based on the likelihood of that input belonging to each class. The decision is made by considering both prior probabilities and the likelihood of observed features given each class. In the context of handwritten digit recognition, the features could be pixel values or higher-level descriptors extracted from the digit images. Similar to kNN classifier, I brought both the handwritten function and python function to determine the difference between their performances.

3.3. Decision Tree

The Decision Tree Classifier is a versatile and interpretable machine learning algorithm widely used for both classification and regression tasks. Its appeal lies in its ability to create a structured decision-making process, resembling a tree-like flowchart, making it intuitively understandable. A Decision Tree is composed of nodes that represent decision points and leaves that correspond to class labels or regression values. Each internal node tests a specific feature, and the branches emanating from the node represent the possible outcomes of the test. The final predictions are made by traversing the tree from the root to a leaf node based on the features of the input data. While Decision Trees offer interpretability and flexibility, they may be prone to overfitting, capturing noise in the training data. This challenge can be mitigated through techniques such as pruning or by using ensemble methods.

3.4. Random Forest

The Random Forest Classifier is a powerful ensemble learning method that leverages the strength of multiple Decision Trees to improve predictive accuracy and robustness. It addresses the limitations of individual trees, such as overfitting, by combining their outputs through a process called bagging (Bootstrap Aggregating). In addition to sampling data, each tree considers a random subset of features at each split. This feature randomization further promotes diversity among the trees. While Random Forests are robust, they may be computationally expensive, especially with a large number of trees.

4. Brief description of the used feature extraction techniques

4.1. Mutual Information

Mutual information is a statistical measure used to quantify the amount of information that one random variable contains about another random variable. The mutual information I(X;Y) between two random variables X and Y is a measure of the amount of information gained about one variable by observing the other. It is defined mathematically as:

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$$

where p(x,y) is the joint probability mass (or density) function of X and Y, and p(x) and p(y) are the marginal probability mass (or density) functions.

In the context of feature extraction for classification, mutual information is used to evaluate the dependence between each feature and the target variable. Features with high mutual information scores are considered more informative for predicting the target variable.

Considerations: Mutual information is particularly useful when dealing with non-linear relationships and dependencies between features and the target variable. It is applicable for both discrete and continuous variables. The choice of the number of features to select (k) is often based on experimentation and model performance evaluation. See Fig. 1.

4.2. Pearson Correlation

Pearson correlation is a statistical measure that quantifies the linear relationship between two continuous variables. In the context of feature extraction for machine learning, Pearson correlation can be used to evaluate the linear dependence between each feature and the target variable (class labels). It helps identify features that exhibit a strong linear correlation with the target, making them potentially informative for classification tasks. It is calculated as the covariance of the two variables divided by the product of their standard deviations. The formula for Pearson correlation between variables X and Y is:

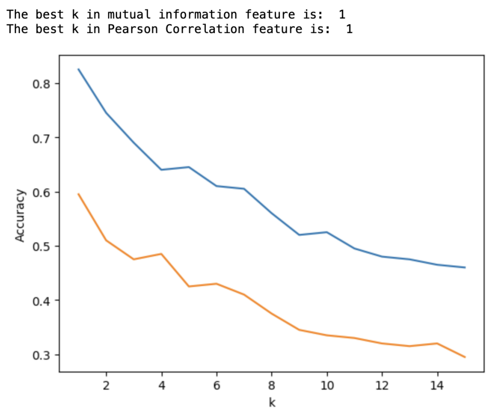
$$r = \frac{cov(X, Y)}{\sigma_X \sigma_y}$$

where cov(X,Y) is the covariance between X and Y, and σ_X and σ_X are the standard deviations of X and Y, respectively.

5. Initialize Parameters

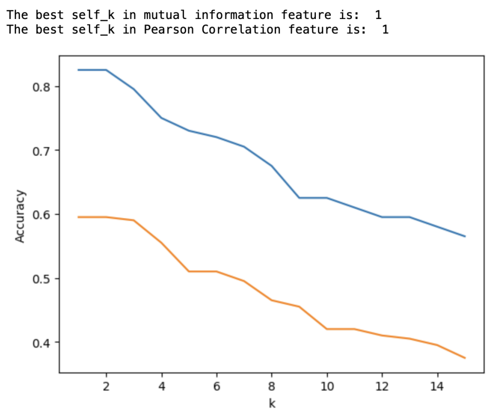
5.1. k

To set the value of k in KNN classifier, I do classification for some 'k's and save the k, which gives the best performance. See Fig. 2 and 3.



```
The best k in mutual information feature is:  1
The best k in Pearson Correlation feature is:  1
```

Figure 2: k in python built-in k-NN function



```
The best self_k in mutual information feature is:  1
The best self_k in Pearson Correlation feature is:  1
```

Figure 3: k in our k-NN function

According to the plot, the best k is 1 for 50 PCA components and final 10 features. Moreover, for further classifications I also found that the best k is 1.

5.2. Bandwidth

To set the value of bandwidth in Parzen Window classifier, I do classification for some bandwidths and save one, which gives the best performance.

6. Results

In this section, the accuracy of each pair of the four classifiers and two feature extraction techniques on the Arabic digit recognition is presented. Moreover, the KNN classification technique have a parameter that needs to be adjusted, which is discussed below.

6.1. Accuracy

Table 1 shows the accuracy of classifier/feature pairs on data of ORL dataset for 50 PCA components and final 10 features.

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation
Self Written k-NN	82.50 %	59.50 %
Python k-NN	82.50 %	59.50 %
Parzen Window	82.50 %	62.00 %
Self Written Bayes	69.50 %	51.00 %
Python Bayes	69.50 %	51.00 %
Decision Tree	43.00 %	27.00 %
Random Forest	78.50 %	56.00 %
Average	72.57 %	52.29 %

Table 1: The accuracy of classifier/feature pairs for 50 PCA component and 10 final feature with 200 trainset and 200 testset.

According to the Table 1, Mutual Information is better than Pearson Correlation as feature extractor with the average of 72.57%. Moreover, among classifiers, k-NN and Parzen Window show better result with the accuracy of 82.50%. Finally, the functions that I wrote work the same as python built-in function.

7. Check Robustness

To comprehensively analyze and assess the robustness of the model, additional experimentation has been conducted. This includes introducing noise to the dataset, varying the number of Principal Component Analysis (PCA) components, and exploring different training sample sizes. The purpose of these supplementary investigations is to evaluate how well the model generalizes to noisy data, understand the impact of different PCA representations, and assess the model's performance across varying training set sizes. These analyses aim to provide insights into the model's resilience and its ability to adapt to different conditions, contributing to a more thorough understanding of its overall effectiveness.

7.1. Different PCA Components

To check the effect of the feature size after PCA, I classify data with different PCA components from 30 to 70. See table 2 and 3 and fig. 4 to 7.

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation	No Feature Extractor
Self Written k-NN	82.50 %	67.00 %	86.50 %
Python k-NN	82.50 %	67.00 %	86.50 %
Parzen Window	82.50 %	68.00 %	86.50 %
Self Written Bayes	69.50 %	53.50 %	66.00 %
Python Bayes	69.50 %	53.50 %	66.00 %
Decision Tree	43.00 %	31.00 %	44.50 %
Random Forest	78.50 %	61.50 %	85.50 %
Average	72.57 %	57.38 %	74.5 %

Table 2: The accuracy of classifier/feature pairs for *pca_min* = 30

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation	No Feature Extractor
Self Written k-NN	82.50 %	58.50 %	87.00 %
Python k-NN	82.50 %	58.50 %	87.00 %
Parzen Window	82.50 %	59.50 %	87.00 %
Self Written Bayes	69.50 %	49.50 %	63.50 %
Python Bayes	69.50 %	49.50 %	63.50 %
Decision Tree	43.00 %	28.00 %	46.50 %
Random Forest	78.50 %	52.00 %	78.00 %
Average	72.57 %	50.5 %	73.21 %

Table 3: The accuracy of classifier/feature pairs for *pca_max* = 70

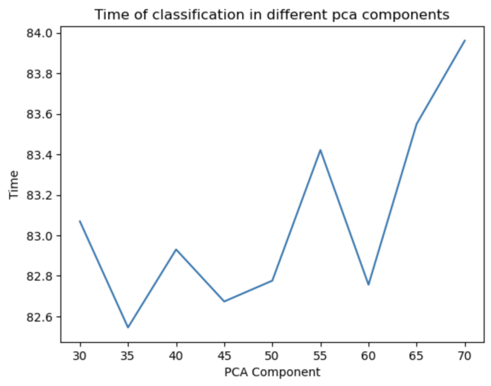


Figure 4: Duration of classification

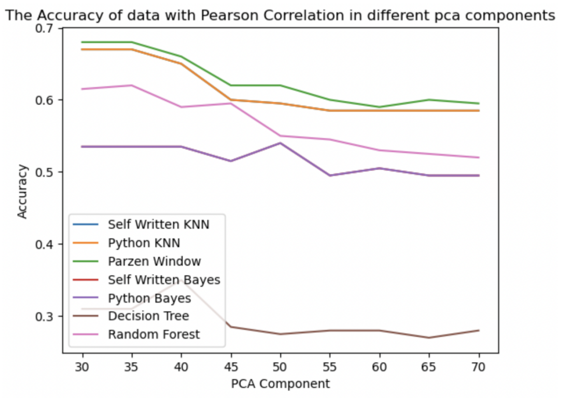


Figure 6: Pearson Correlation

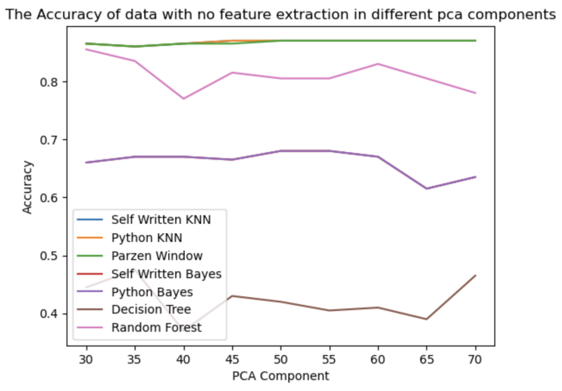


Figure 5: No Feature Extraction

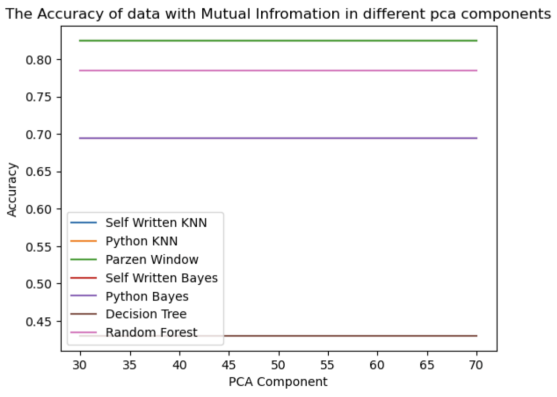


Figure 7: Mutual Information

According to fig 4 that shows the time during different classifications with different PCA components, we can see that the more components increase the more time increases. Additionaly, the accuracy decreases when the components increase. However, in the diagram ,which mutual information features are shown we can see that there is no significant change in the accuracy. One reason for that can be the small number of features (10 features) or the independency of this feature extractor with PCA.

7.2. *Different Training Sample Size*

In this part, the data is classified with different sample sizes from 1 to 9 images per person for training and 9 to 1 images per person for testing in order to find the effect of training sample size on accuracy. See fig. 8 and 9 and table 4 and 5.

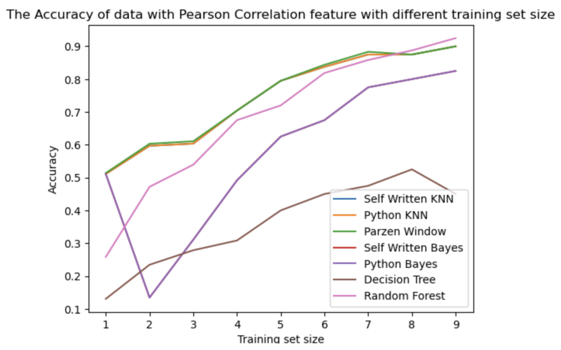


Figure 8: The effect of sample size on data with Pearson Correlation feature extraction

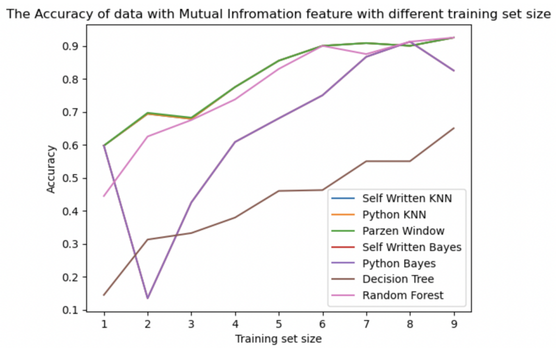


Figure 9: The effect of sample size on data with Mutual Information feature extraction

Due to the diagrams, The more our testset size increases the more accuracy increases in a way that in overall we have the highest accuracy in training with 9 sample per person and testing with one sample per person, however, in some classifier/ feature extractors there is a drop in this sample size. Moreover, we can see that Parzen Window and k-NN demonstrate the best accuracy among our classifiers.

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation
Self Written k-NN	53.89 %	36.39 %
Python k-NN	53.89 %	36.39 %
Parzen Window	53.89 %	36.39 %
Self Written Bayes	53.89 %	36.39 %
Python Bayes	53.89 %	36.39 %
Decision Tree	16.39 %	8.06 %
Random Forest	42.78 %	23.89 %
Average	46.95 %	30.56 %

Table 4: The accuracy of classifier/feature pairs for sample size of 1 image per person for training set

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation
Self Written k-NN	92.50 %	82.50 %
Python k-NN	92.50 %	82.50 %
Parzen Window	92.50 %	82.50 %
Self Written Bayes	80.00 %	72.50 %
Python Bayes	80.00 %	72.50 %
Decision Tree	60.00 %	40.00 %
Random Forest	90.00 %	77.50 %
Average	83.93 %	72.86 %

Table 5: The accuracy of classifier/feature pairs for sample size of 1 image per person for training set

7.3. *Add noise*

To determine the robustness of the model against noisy images, the salt and pepper noise is added to 10% of the features of the test samples. See table 6.

Feature Extractor/ Classifier	Mutual Information	Pearson Correlation
Self Written k-NN	4.00 %	2.00 %
Python k-NN	4.00%	2.00 %
Parzen Window	4.50 %	2.50 %
Self Written Bayes	2.50 %	2.00 %
Python Bayes	2.50 %	2.00 %
Decision Tree	3.50 %	4.00 %
Random Forest	2.50 %	4.00 %
Average	3.35 %	2.64 %

Table 6: The accuracy of classifier/feature pairs on noisy testset

According to the table, an interesting observation is that the model trained on data with Mutual Information features consistently demonstrates superior performance compared to the model trained on data with Pearson Correlation features. Furthermore, comparing the accuracy obtained using the self-written functions and the corresponding Python functions, it is evident that both implementations yield identical accuracy results. Notably, when introducing noise to the dataset, the model's accuracy experienced a substantial decrease. This decline suggests that the model is sensitive to noise and highlights a potential vulnerability in its robustness.

8. Refrence

1. https://cam-orl.co.uk/facedatabase.html