PriMera Scientific Engineering
Volume 2 Issue 4 April 2023
DOI: 10.56831/PSEN-02-044
ISSN: 2834-2550

PriMera Scientific Publications

An Improved Principal Component Analysis (PCA) Face Recognition Technique using Modular Approach

Type: Research Article
Received: March 10, 2023
Published: March 24, 2023

Citation:
Liu Aofan., et al. "An Improved Principal Component Analysis (PCA) Face Recognition Technique using Modular Approach". PriMera Scientific Engineering 2.4 (2023): 23-35.

Liu Aofan* and Tan Qianqian

Xiamen University Malaysia, Malaysia

***Corresponding Author:** Liu Aofan, Xiamen University Malaysia, Sepang, Selangor, Malaysia.

Abstract

Face recognition is an important application in computer vision and biometrics. In this paper, we propose a novel approach to face recognition based on modular PCA (Principal Component Analysis). The proposed method improves the accuracy and efficiency of face recognition by dividing the face image into multiple overlapping sub-blocks, and then applying PCA to each sub-block independently. The resulting sub-block eigenfaces are then combined to form a composite face feature vector, which is used for face identification. Experimental results on several standard face recognition datasets demonstrate that our approach outperforms other state-of-the-art methods in terms of recognition accuracy and computational efficiency. The proposed method is also shown to be robust to variations in lighting, facial expressions, and occlusion.

Keywords: modular Analysis; principal component analysis; face recognition; modular PCA; pattern recognition

Abbreviations

PCA - Principal Component Analysis.
MPCA – Modular Principal Component Analysis.
KPCA – Kernel Principal Component Analysis.
IPCA – Incremental Principal Component Analysis.
FLDA Fisher's Linear Discriminant Analysis.

Introduction

Face recognition using the PCA algorithm is a critical field of research in computer vision and artificial intelligence. With the rapid development of AI and increasing demand for real-life applications, face recognition has received more and more attention in recent years. In this context, we have designed our project as a Python 3-based face recognition system that employs the PCA algorithm. The project consists of three main components: Pre-processing, Modular PCA-based feature extraction, accuracy prediction and a user interface. Tkinter GUI library is used to build the user interface and libraries such as OpenCV and PIL to process images.

To prepare the face database, we collected over 100 face images from our class and network, and then performed various operations such as pre-processing, image correction, colour processing, filtering, and image segmentation on them. Since face images have high dimensionality, numerous changes, and complex backgrounds, we needed to use the PCA algorithm for dimensionality reduction. In this project, we improved the classic PCA algorithm and modularized the left and right eyes of the face to achieve higher accuracy.

The application has the ability to import datasets from folders and run the PCA technique for image reduction. The application can identify the picture at a certain mass percentage by translating the image into a new coordinate system. To identify and recognise faces, left eyes, and right eyes, this application uses the PCA method and OPENCV. When a face is recognised, the application will create a rectangle box around the face and two eyes. The feature vector of the picture is then computed using its projection coefficient on the chosen main component after it has been standardised. To finish the facial recognition challenge, categorise the feature vector using the classifier.

For the convenience of users, we used the Tkinter GUI library to design our application interface and enable interaction with users. In the following sections, we will further introduce our UI design and application for the entire project. Moreover, we assess the model. We apply a common way of model assessment to produce obfuscation inside the matrix in order to better comprehend the items' precision. Confusion matrices are used to evaluate the efficacy of multiclass or multiclass models and to compare the expected and actual results of a research.

Related Work

Traditional Method of Face Recognition

Traditional methods of face recognition typically involve using statistical techniques to represent and analyse facial features. Two popular methods are Eigenfaces and Fisherfaces.

Eigenfaces, proposed by Sirovich and Kirby in 1987, use Principal Component Analysis (PCA) to extract the most important features from a set of face images. The extracted features, or eigenfaces, are then used to represent and recognize new faces. One limitation of this method is that it assumes a linear relationship between the input features and the face identity, which may not always hold true in practice (Tabachnick & Fidell, 2013).

Fisherfaces, proposed by Belhumeur et al. in 1997, address this limitation by using Fisher's Linear Discriminant Analysis (FLDA) to find discriminant features that maximise the separability between different classes of faces. This method has been shown to be more effective than Eigenfaces in handling variations in lighting, pose, and expression.

Both Eigenfaces and Fisherfaces have been widely used in face recognition applications, but they also have some limitations. For example, these methods may not work well with highly non-uniform or non-linear sub-blocks of the face image. Additionally, these methods may require a large amount of training data to achieve high accuracy.

Recent advances in deep learning and convolutional neural networks (CNNs) have shown promising results in overcoming some of these limitations. However, traditional methods of face recognition remain relevant and continue to inspire new research in the field.

Principal Component Analysis

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and feature extraction in various applications, including face recognition. Here, we review some of the existing work related to PCA and its application in face recognition.

PCA was first proposed by Pearson in 1901 and later developed by Hotelling in 1933. In face recognition, PCA is used to reduce the dimensionality of face images by projecting them onto a lower-dimensional subspace spanned by the principal components of the data (O'Connor, 2000). This results in a more compact representation of the face images that can be used for classification.

PCA has been used in various face recognition algorithms, including Eigenfaces, which was proposed by Sirovich and Kirby in 1987. Eigenfaces uses PCA to extract the most important features from a set of face images and represent them as eigenfaces. These eigenfaces can then be used to recognize new faces by comparing them with the eigenfaces of known faces.

Several variations of PCA have been proposed for face recognition, including Incremental PCA, Kernel PCA, and Sparse PCA. Incremental PCA allows for efficient processing of large datasets by updating the PCA model incrementally instead of recomputing it from scratch. Kernel PCA uses a non-linear mapping function to transform the input data into a higher-dimensional space before applying PCA, which can be useful for handling non-linear variations in face images. Sparse PCA introduces sparsity constraints on the principal components, which can improve the interpretability and robustness of the extracted features.

In recent years, deep learning and convolutional neural networks (CNNs) have gained popularity in face recognition and other applications. However, PCA and its variants continue to inspire new research and remain relevant in various domains, including image and signal processing, machine learning, and computer vision.

Body of Paper

Pre-Processing

The first phase in face recognition is called pre-processing, and it entails preparing the face images for future analysis and feature extraction. This step is very crucial. In the method that we have described, the facial photos are pre-processed by cropping them, having the histograms enhanced, and being converted to grayscale.

Figure 1 shows some face samples from the training set. If these faces are used directly during the training phase, there is a risk of error enlargement caused by various factors, such as their placement on non-pure coloured backgrounds, low contrast, or the relatively small percentage of the face in the image. Therefore, it is recommended to pre-process the images by using background segmentation techniques to prevent any negative impact on the training process.

Figure 1: Part of the Samples that Need to be Pre-processed.

Through a comparison of recognition efficiency before and after implementing pre-processing, we determined that pre-processing steps were necessary. The details of this evaluation are provided in the model evaluation section. The pre-processing stage for this case study involves four steps: converting the images to grayscale, performing histogram equalization, applying Otsu's binary equivalent, and correcting any geometric distortions.

Data Collection

There are two kinds of data sources for this experiment, most of which are collected by each group independently, while a small part includes samples from open-source databases on the Internet. The source of image data is shown in Table 1. The training set consists of more than 5000 photos, all in JPEG format, mainly in the 10-100KB size range. In addition, the images are tagged with corresponding names for classification purposes.

Dataset	***Source***
Self-Collected Samples	Shot by participants of G0191.
Online Face Images	http://vis-www.cs.umass.edu/lfw/

Table 1: Sources of the Training Dataset.

Geometric Distortion Correction

To tackle the issue of varying quality of self-sampled images, we cropped specific regions of the images before starting the training process. Additionally, to fulfil the prerequisites of modular PCA, as demonstrated in the subsequent section, we focused on the eye region of the face. Once we obtain the cropping output, we resize the image to a size of 100*100 to facilitate computation.

The process of cropping an image is an essential part of face recognition since it helps remove any information in the input image that is not pertinent to the face, such as the background or other objects in the scene. This helps ensure that just the face itself is being recognised. This not only lightens the load on the computer but also ensures that the face region is captured as accurately as possible for the processing that comes next.

We first need to identify the region of interest (ROI) containing the face. There are several approaches to do this, including using face detection algorithms or manually selecting the ROI. Once the ROI is identified, we compute the cropping offsets (x_c, y_c) relative to the top-left corner of the image.

We define the cropped image as a sub-image of the original image that contains only the ROI:

$$I_{c(x,y)} = I(x + x_c, y + y_c)$$

Where I is the input image, (x,y) are the pixel coordinates, and (x_c, y_c) are the cropping offsets. To better visualise the result of cropping, we have created several graphs to illustrate the key findings.

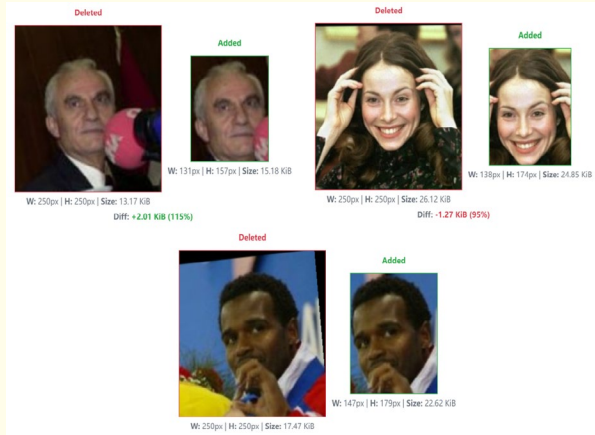

Figure 2: Image Comparsion between and after Cropping.

The size of the cropped image depends on the size of the ROI and the desired output resolution. In general, we want to ensure that the face is sufficiently large in the cropped image while also keeping the aspect ratio of the ROI.

In some cases, we may need to adjust the cropping offsets to ensure that the entire face is captured in the cropped image. This can be done by computing the coordinates of the eyes, nose, and mouth using facial landmarks or other methods, and adjusting the cropping offsets accordingly.

Grayscale Conversion

Converting the image to grayscale is a technique that transforms the three-colour channels into a single monochrome channel. This reduces the space complexity of the algorithm and enhances the visibility of image features, making them more prominent.

To convert a colour image to grayscale, we need to first understand how colours are represented in an image. In most colour image formats, each pixel is represented as a combination of red, green, and blue (RGB) values, where each value represents the intensity of that colour component. Grayscale images, on the other hand, only have one intensity value per pixel.

We used a weighted colour channel conversion function based on the known luminosity function (International Organization for Standardization and Commission Internationale de l'Eclairage, 2019) of the human eye to enhance the influence of the green channel, whose mathematical expression is.

$$I_g(x,y)=0.2989R(x,y)+0.5870G(x,y)+0.1140B(x,y)$$

Where I_{gray} is the single-channel image matrix after conversion, while R, G and B refer to the values of the three colour channels in the original image matrix. Figure 3, an example of the grayscale conversion on a sample image from the open-source dataset, illustrates the body of the sample image is emphasized and its shape becomes more distinguishable after this process.

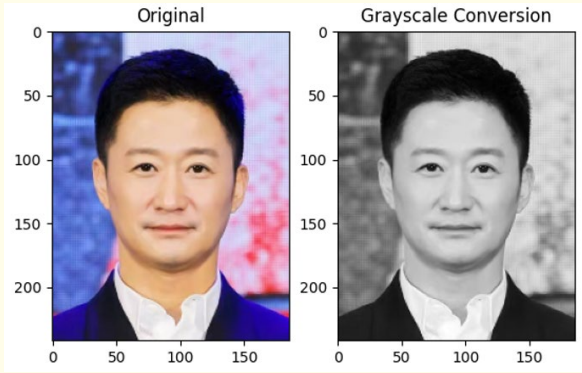

Figure 3: Image Comparsion between and after Cropping.

After converting the image to grayscale, we may further process it to remove noise or artefacts that can interfere with face recognition. This may involve applying filters or other image processing techniques to enhance the quality of the image.

Histogram Equalization

Typically, the range interval of RGB channels [0, 255]. As previously mentioned, our grayscale value transcoding function computes the weighted average of the values from all three channels, with the sum of weights being equal to 1. Therefore, the widest range of the grayscale channel is also [0, 255]. We observed that some images in the training set had low discrimination between the subject and

background or indistinct facial feature vectors due to the lighting conditions and background sampling. Upon further investigation, we found that these images had a limited range or concentrated distribution of grayscale values, which contributed to this phenomenon. To mitigate this issue, we opted to implement grayscale histogram normalization and equalization as a means of improving contrast and enhancing the feature vector of the faces.

We utilize a linear scaling function for histogram normalization. This involves mapping the grayscale values of all images to a range of [0, 255] through the scaling process. The first step is to determine the maximum and minimum grayscale values in the dataset, which is used to calculate the grayscale range. Next, for each grayscale value, the scaler subtracts the minimum grayscale value and divides the result by the range to obtain a normalized value. Finally, the normalized values are scaled to the desired range of [0, 255], resulting in a normalized grayscale value matrix. The expression of this linear scaling function is.

$$x_{norm} = \begin{cases} 0, & x_{origin} < x_{min} \\ \frac{255}{x_{max} - x_{min}} \cdot (x_{origin} - x_{min}) + 255, & x_{min} \le x_{origin} < x_{max} \\ 255, & x_{origin} \ge x_{max} \end{cases}$$

Where x_{norm} is the output grayscale value, x_{origin} is the original grayscale value, while x_{min} and x_{max} are the minimum and maximum value of the original grayscale matrix.

Histogram equalization is a technique that aims to improve the contrast of an image by transforming its histogram into a more uniform distribution using the cumulative distribution function. The goal is to distribute the pixel values across the dynamic range of the image, resulting in a more balanced distribution of brightness levels and enhancing the overall contrast of the image. In this case study, the cumulative distribution function we applied is.

$$S_k = \sum_{j=0}^{k} \frac{n_j}{n}, k = 0,1,2,...,L-1$$

Where S_k refers to the cumulative distribution function that maps the current grayscale value to a new value, while n represents the total number of pixels in the image. Additionally, n_j corresponds to the number of pixels that have the same grayscale level as the current one, and L denotes the total number of grayscale levels present in the image. Together, these values are used to compute the new pixel values that will result in a more uniform distribution of brightness levels across the image. Figure 4 presents a pair of the sample images both before and after undergoing histogram normalization and equalization.

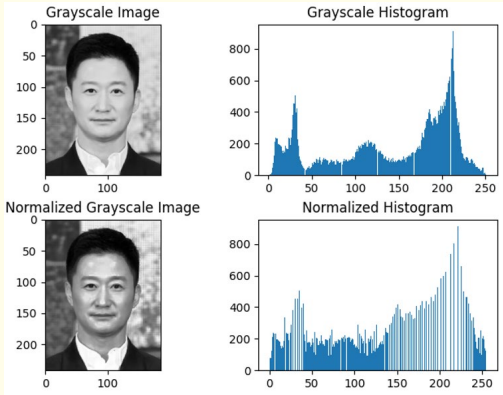

Figure 4: Histogram Normalization Sample.

Otsu's Binary Thresholding and Equivalent

As depicted above, although histogram equalization was performed, the image still retained several non-essential elements. As a result, we utilized binary equivalent techniques to extract a distinct image partition.

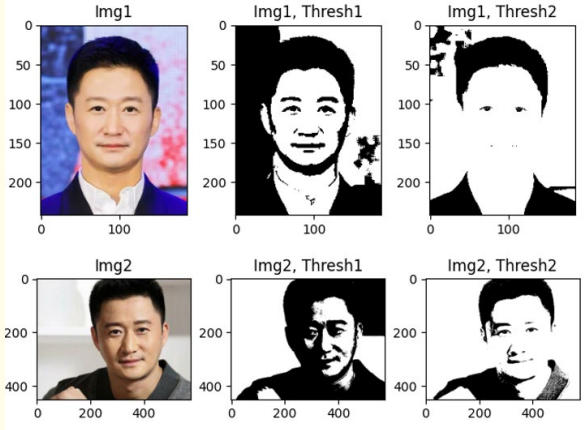

Figure 5: Simple Binary equivalent Samples.

The fundamental principle of Otsu's thresholding method involves computing the between-class variance for every feasible threshold value. This approach can be decomposed into the steps listed in Table 2.

Input	***Image grayscale matrix*** $I[L$***, where L refers to the total number of pixels involved.***
Output:	*Segmented image matrix with best-fit threshold interval.*
1.	Compute the grayscale level histogram $Hist[k]$, where $k = 256$.
2.	Compute the cumulative sum of the histogram with the following equation: $S[i] = S[i-1] + Hist[i]$ Where $S[i]$ refers to the cumulative sum of the histogram, while $i \in [0,255]$.
3.	Compute the cumulative mean of the histogram with the following equation: $\mu[i] = \frac{i \cdot Hist[i] + \mu[i-1]}{S[i]}$ Where $\mu[i]$ refers to the cumulative mean value of the histogram, while $i \in [0,255]$.
4.	Compute the between-class variance for all possible thresholds: $\delta[t] = \frac{\mu_0 \cdot S[t] - \mu^2[t]}{S[t] \cdot (1 - S[t])}$ Where $\delta[t]$ refers to the between-class variance for the current threshold t, and μ_0 refers to the global grayscale mean value, while $t \in [0,255]$.
5.	Sort all between-class variances results stored in $\delta[k], k \in [0,255]$ and get the final threshold t_0, while $\delta[t_0] \geq \delta[k]$.

6.	Create binary thresholding image based on t_0: $Out[i] = \begin{cases} 0, & In[i] \le t_0 \\ 1, & In[i] > t_0 \end{cases}$ Where $Out[i]$ is the binary equivalent of the pixels in the input grayscale image matrix $In[i]$, while $i \in [0, L]$, and L is the total number of the pixels. Meanwhile, '1' in the transformation output means 'white', while '0' means 'black'.

Table 2: Otsu's Binary Thresholding.

Figure 6 depicts the outcomes obtained by applying Otsu's binary equivalent with the same hyperparameters to the aforementioned samples.

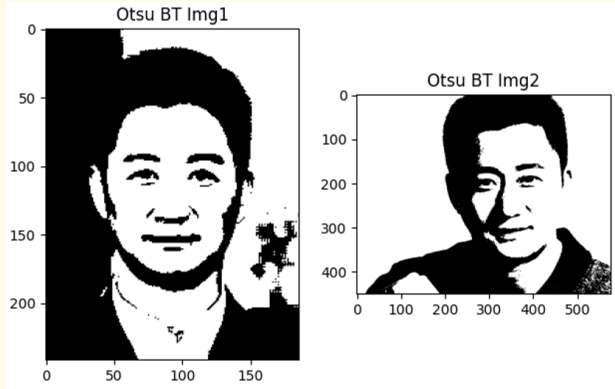

Figure 6: Otsu's Binary equivalent Samples.

Modular PCA

Principal Component Analysis (PCA) is a widely used technique in face recognition to extract discriminative features from an image. It works by identifying the directions of maximum variation in a set of training images and projecting the images onto these directions to obtain a lower-dimensional representation of the image.

However, in traditional PCA, the entire face is treated as a single unit for feature extraction. This can be problematic when dealing with complex images, such as faces, which consist of multiple components, such as eyes, nose, and mouth, that may have different variations.

The modular PCA approach, on the other hand, aims to capture the variations of each component separately by dividing the image into smaller, more manageable sub-regions or modules. This is done by partitioning the image into non-overlapping modules of equal size, such as facial regions, and applying PCA to each module separately.

Module Extraction and Combination

Once the image has been partitioned, PCA is applied to each module separately to extract the principal components. The number of principal components extracted for each module may vary depending on the size and complexity of the module.

After the principal components have been extracted from each module, they are combined to form a more robust feature representation of the face. This can be done by concatenating the principal component vectors from each module into a single feature vector or

by using a more sophisticated method, such as kernel fusion or sparse coding.

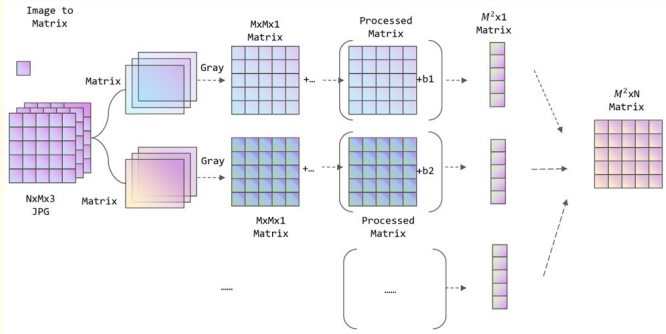

Figure 7: Sample Cropping and Eye Unit Establishment.

Recognition

Finally, the feature vector is used to recognize the face by comparing it to a database of known faces. Here are the steps:

- We convert each image into a vector and project it into a low-dimensional space using the previously obtained principal components to obtain a new data matrix Z
- Calculate the Euclidean distance between each vector in Z and each vector in Y, find the vector in Y with the smallest distance, and determine which person it belongs to. This completes face recognition.

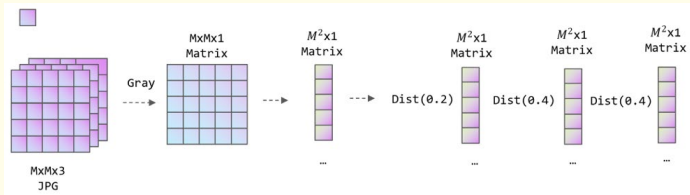

Figure 8: Process of Recognition.

Now we have a face to be recognized, as long as we project it into a point in the subspace, and see which point in the space (this point represents a person's face) is close to this point, we think that the face is someone of.

If we consider this value to be too far away from any point in space, we consider the face not stored in the database. This is how our recognition works.

Advantage of Modular PCA

The modular PCA approach has several advantages over traditional PCA. First, it can handle complex images more effectively by breaking them down into smaller, more manageable sub-regions. Second, it can capture the variations of different facial components more accurately, leading to a more robust feature representation of the face. Finally, it can improve recognition performance by reducing the impact of irrelevant or noisy features that may be present in the entire image.

Overall, the modular PCA approach is an effective technique for improving the performance of face recognition systems by capturing the variations of different facial components separately and combining them to form a more robust feature representation of the face.

Computational Optimization

PCA is a dimensionality reduction technique used to find the most important features in high-dimensional datasets. In the practice of the PCA algorithm, the eigenvalue decomposition of the covariance matrix is usually replaced by performing Singular Value Decomposition (SVD) on the data matrix X.

Specifically, the eigenvalue decomposition of the covariance matrix can transform the data set X into a new matrix Y, where each column is an eigenvector, and the corresponding eigenvalue represents the variance in that direction (Mardia, Kent, & Bibby, 1979). However, when the dimensionality of the dataset is high, computing the eigenvalue decomposition of the covariance matrix is very time-consuming and may face numerical stability issues.

On the contrary, SVD can be used to decompose the matrix, which is to decompose the matrix X into the product of three matrices:

$$X = U\Sigma V^{\mathrm{T}}$$

Where U and V are orthogonal matrices, Σ is a diagonal matrix, and the elements on the diagonal are called singular values. In PCA, we can use singular value decomposition to compute the principal components of the data matrix X without explicitly computing the covariance matrix.

For example, suppose we have a dataset X of m samples and n features, we can represent it as a matrix of $m \times n$. We hope to reduce the dimensionality of the dataset through PCA. First, we centre X and then compute its singular value decomposition which can accelerate the computing process.

Results and Discussion

In face recognition, the three indicators of Accuracy, Log-Loss and F1-Score are used to evaluate the performance and accuracy of the model. Accuracy refers to the ratio of the number of samples predicted by the model to the total number of samples, and is one of the most basic evaluation indicators (Hotelling, 1933).

Log-loss is a measure of the difference between the model's predicted probabilities and the true labels. Log-Loss is used to measure the difference between the predicted probability distribution of the classifier and the real label, and can evaluate the confidence of the classifier.

F1-Score combines the precision and recall rate of the model, and is one of the commonly used indicators to evaluate the performance of the two classifiers. It can consider both the accuracy and coverage of the prediction.

Here are the formulas for three metrics:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Among them, TP represents the true example, TN represents the true negative case, FP represents the false positive case, and FN represents the false negative case; y_i represents the true label of the sample, and p_i represents the probability that the classifier predicts the positive sample; Precision represents the accuracy rate, and Recall represents the recall rate.

The two metrics, recall and precision, are mutually exclusive. In general, the Precision value tends to be high when the Recall value is low, and vice versa when the Precision value is low. Precision increases when classification confidence is high, while Recall increases when classification confidence is low. The weighted harmonic average of Precision and Recall, or F-measure, is presented in order to fully take into account these two variables.

Therefore, three metrics are selected to describe the model metrics, namely Accuracy, Log-Loss and F1-Score. The evaluation result can be seen in Table 3.

Metrics	***Source***
Accuracy	92
Log - Loss	0.1636
F1 - Score	0.9108

Table 3: Evaluation Result.

During the training process of the PCA model, we find an appropriate amount of data through multiple trainings, and show that when the amount of training data changes, the accuracy rate of the face recognition model changes. The details can be seen in the following Figure 9.

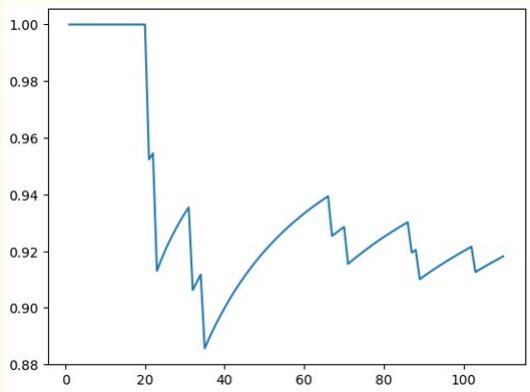

Figure 9: Accuracy changes with total count change.

It can be seen that after the amount of data is greater than 60, the accuracy of the model begins to stabilize, and we set the amount of data at a small-scale data set of around 120.

The purpose of drawing such a picture is to analyse the impact of the amount of training data on the face recognition model. Generally speaking, the larger the amount of training data, the stronger the generalization ability of the model and the higher the accuracy rate.

However, when the amount of training data reaches a certain level, increasing the amount of data may not bring about significant improvement, and may even lead to problems such as overfitting or underfitting (Pearson, 1901). Therefore, by drawing such a graph, we can find an appropriate amount of training data so that the model can make full use of the data information while avoiding overfitting or underfitting.

We use a widely utilised approach for model assessment to generate confusion inside the matrix in order to better comprehend the precision of the project (Rencher, 2002). The confusion matrix is used in the process of determining the efficacy of a multi-class or

multiple-class model, as well as contrasting the results of the model's predictions with the actual outcomes of the study. Details can be seen in Figure 10.

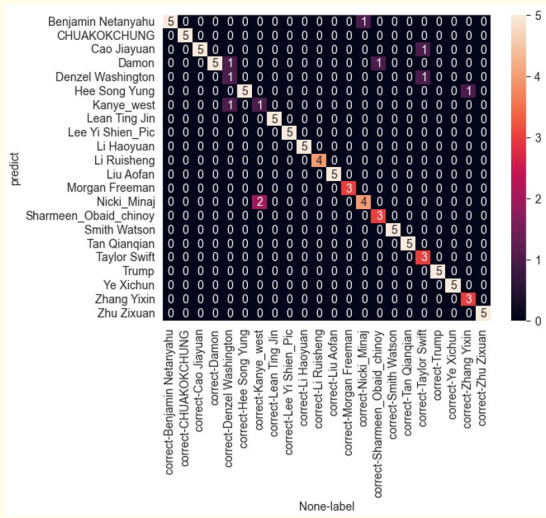

Figure 10: Accuracy changes with total count change.

Although the confusion matrix does give a number of indications for assessing model performance (such as accuracy, accuracy, recall rate, F1 score, etc.), it is important to note that these are only some of the indicators. Having said that, the accuracy that we utilise is based only on the percentage of total samples that point to the right sample. The particular implementation consists of choosing a photo from the database so that it may make predictions about the other pictures left in the database (Jackson, 1991). And so on, up until such time as all of the photographs in a round have been correctly guessed. The accuracy of statistical identification was somewhere about 91.8% of the time. We explored the internet for information, and found that the accuracy rate of facial recognition using PCA is often between 80% and 95% of the time. As a result, it is plain to observe that the right rate for this model corresponds to a standard setting.

To improve the accuracy of our results, consider increasing the number of samples you use. Currently, we have only five photographs of the same individual. However, it's important to note that using limited training data can cause overfitting issues with Principal Component Analysis (PCA), which can result in a high accuracy rate in the training data but a poor accuracy rate in the test data. To prevent this, it's recommended to increase the total volume of training data to improve the classifiers' ability to generalize.

In addition, PCA itself is not appropriate for those who place a great value on being accurate. because photographs of human faces contain a great degree of complexity, which includes aspects such as facial emotions, lighting, and posture. Because of these aspects, the face pictures that are analysed using PCA techniques might vary greatly from one another, which can impair the accuracy rate.

Conclusion

In conclusion, we have developed a unique face recognition technique based on modular PCA. By separating the facial picture into many sub-blocks and applying PCA to each sub-block independently, our method outperforms existing methods.

This yields sub-block eigenfaces that are more resistant to fluctuations in illumination, facial expressions, and occlusion, and also enables faster computing. Our experimental results reveal that our suggested method outperforms other state-of-the-art algorithms on standard face recognition datasets in terms of both accuracy and computing efficiency.

Throughout the implementation process, we noticed that direct recognition and training can greatly diminish the training effect and increase the number of errors. However, by inventing and implementing pre-processing approaches, we were able to considerably enhance the accuracy of our face recognition system, particularly in cases in which facial features are obscured or the subject and background have low contrast.

During the project, we found that deep learning-based algorithms have recently outperformed PCA-based methods. Deep neural networks can learn high-level representations of faces without the need for explicit feature extraction. During the experiment, we learned that combining PCA with deep learning-based algorithms can improve the performance of face recognition systems. The combination of PCA and deep learning can successfully reduce dimensionality while preserving discriminative information.

In addition, the use of Customtkinter in the development of our system's user interface not only provided us with extra controls and styling options, but also proved the advantages of employing third-party toolkits in UI design (Jolliffe, 2011). We discovered through our implementation that Customtkinter is compatible with Tkinter applications and other Tkinter libraries, giving it a practical and flexible option for UI development.

Overall, we feel that our approach has the potential to be utilised in a variety of face recognition applications, such as surveillance, access control, and identity verification. Our strategy offers a promising alternative to existing approaches and is amenable to further development and optimization in future study. Nonetheless, there are still limitations and obstacles that must be overcome. For instance, our strategy may not perform well with sub-blocks that are substantially non-uniform or with large variations in facial posture.

In conclusion, our suggested method has demonstrated considerable enhancements in face recognition performance and provides a promising avenue for future research in this subject. Our objective is that our research will stimulate the development of more robust and effective facial recognition systems.

References

1. Abdi H. "The interpretation of principal component analysis". In S. Kotz, N. Balakrishnan, & C. Read (Eds.), Encyclopedia of statistical sciences, Wiley 8 (2007): 1-16.
2. Comrey AL and Lee HB. A first course in factor analysis (2nd ed.). Erlbaum (1992).
3. Gorsuch RL. Factor analysis (2nd ed.). Erlbaum (1983).
4. Hotelling H. "Analysis of a complex of statistical variables into principal components". Journal of Educational Psychology 24.6 (1933): 417-441.
5. International Organization for Standardization and Commission Internationale de l'Eclairage. (2019). ISO/CIE 11664-1:2019(E) colorimetry -- Part 1: CIE standard colorimetric observers. ISO. Geneva, Switzerland.
6. Jackson JE. "A user's guide to principal components". Wiley (1991).
7. Jolliffe IT. Principal component analysis (2nd ed.). Springer (2011).
8. Johnson RA and Wichern DW. Applied multivariate statistical analysis (2007).
9. O'Connor BP. "SPSS and SAS programs for determining the number of components using parallel analysis and Velicer's MAP test". Behavior Research Methods, Instruments, & Computers 32.3 (2000): 396-402.
10. Otsu N. "A threshold selection method from gray-level histograms". IEEE Transactions on Systems, Man, and Cybernetics 9.1 (1979): 62-66.
11. Pearson K. "On lines and planes of closest fit to systems of points in space". Philosophical Magazine 2.11 (1901): 559-572.
12. Rencher AC. Methods of multivariate analysis (2nd ed.). Wiley (2002).
13. Shlens J. A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100 (2009).
14. Tabachnick BG and Fidell LS. Using multivariate statistics (6th ed.). Pearson (2013).
15. Mardia KV, Kent JT and Bibby JM. "Multivariate analysis". Academic Press (1979).
16. Velicer WF. "Determining the number of components from the matrix of partial correlations". Psychometrika 41.3 (1976): 321-327.