EE4208: Intelligent System Design

Real-time Facial Recognition with Eigenfaces

Academic Year 2021/22

Siddesh Sambasivam Suseela (U1822929B)

Timothy Toh (U1922117J)

Neo Jin Tee (U1921092E)

Teo Jia Ling (U1822376D)

09 April, 2022

School of Electrical and Electronic Engineering

Nanyang Technological University, Singapore

Contents

List of Figures

1 Introduction

Facial recognition is a feature-based image processing technology that is capable of making analysis and patterns according to an individual's facial contours. With the exponential growth in data and computational power deep learning architectures such as convolutional neural networks have become extremely good in learning spatial features. However, conventional methods such Principle component analysis (PCA) are still being used as a feature extraction technique due to its effectiveness.

The objective of the project is build a real-time facial recognition system using principle component analysis (PCA) and study its effectiveness. Additionally, to gain better insights on performance of the built system, results from deep learning techniques were used for comparative measures.

2 Facial Recognition System Implementation

2.1 Data Collection

The data collection process was conducted in two methods. First, a photo capture script was written to capture the faces of classmates, friends and family. A set of 5 facial images was collection for each person (20 people). Second, to create the "unknown" face class, a simple data crawling script was written to scrape a few images of random faces from the internet.

At the end of the data collection process, a hundred images were collection with the first method and 20 images with the second method. However, to ensure the distribution of the dataset is uniform, the number of images in the unknown class was reduced to six.

2.2 Data preprocessing

To perform the preliminary data preprocessing, OpenCV python package was used for rescaling and gray scale conversion. All the images were rescaled to 300x300 dimensions and converted to gray scale. Facial images are normalised and resampled with similar pixel resolution.

Facial extraction works by extracting the eigenfaces out of the facial images. It should be noted that the face images should be properly aligned first.

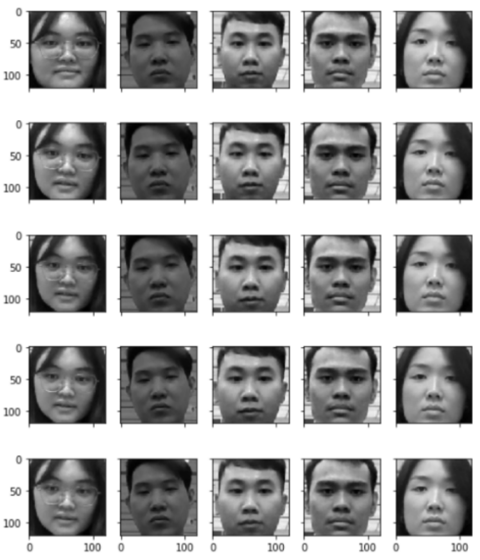


Figure 1: Data Visualisation after rescaling

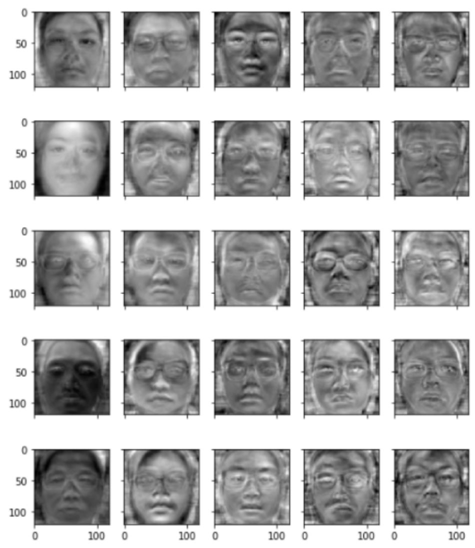


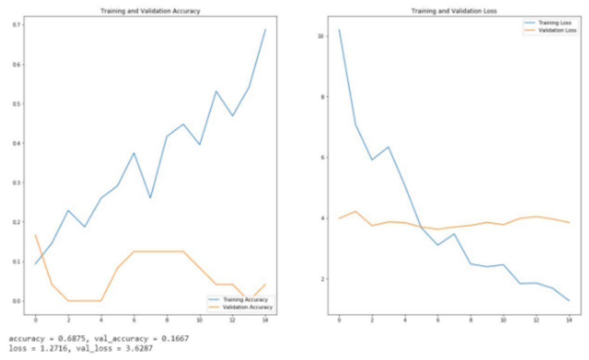
Figure 2: Eigenfaces

2.3 Face Detection

2.3.1 Model selection experiments

2.3.1.1 ResNet-50 A widely used pretrained model ResNet-50 with network trained on more than a million images from the ImageNet database was explored with our imagery dataset. Training and testing dataset was split into a ratio of 80:20. The accuracy and validation is 0.687 and 0.1667 respectively as shown in the figure below.

The model was trained for 15 epochs. After evaluating the training and validation accuracy, it was concluded that the model has resulted in overfitting. To address the overfitting issue, data augmentation can be done to increase the number of training dataset. In addition, other methods such as weight regularization, adaptive learning rates, early stopping with dropout layers can be used to tackle the overfitting problem.



```
accuracy = 0.6875, val_accuracy = 0.1667
loss = 1.2716, val_loss = 3.6287
```

Figure 3: Training and Validation Accuracy for ResNet-50

2.3.1.2 ResNet-10 OpenCV's dnn module supports loading models to address complex deep learning tasks. Therefore ResNet-10 was used from the dnn module which is based on a single shot detector (SSD). It was also fastest among the four models.

2.3.1.3 Multi-Task Cascaded Convolutional Neural Network (MTCNN) MTCNN uses three networks. The first model proposes facial region, the second-bounding boxes and the third proposes facial landmarks. Although the performance of the model was really good, but its quite a lot slower when compared to the other models.

2.3.1.4 Haar Classifier It is trained on Haar-like features. These can be edges or lines, and can be quite powerful when used for detecting objects. Haar-features are good at describing shading and are a very easy way to quickly detect objects. However, it is also prone to false-positive detections and require parameter tuning when being applied for inference/detection.

2.3.2 Selection Remarks

Among the four models, **ResNet-10** was the best fit for the project. As it was relatively accurate in detection and faster inference.

2.4 Principle Component Analysis (PCA)

Principal Component Analysis (PCA) enables the reduction of dimensions as a practical way to reduce the number of features within a large dataset without losing its significance. In this case, PCA introduces the development of eigenface technique for extraction of significant images from an imagery dataset. Eigenfaces are the eigenvectors of the covariance matrix of the images.

1. Standardize the dataset.
2. Calculate the covariance matrix for the features in the dataset.
3. Calculate the eigenvalues and eigenvectors for the covariance matrix.
4. Sort eigenvalues and their corresponding eigenvectors.
5. Pick k eigenvalues and form a matrix of eigenvectors.
6. Transform the original matrix.

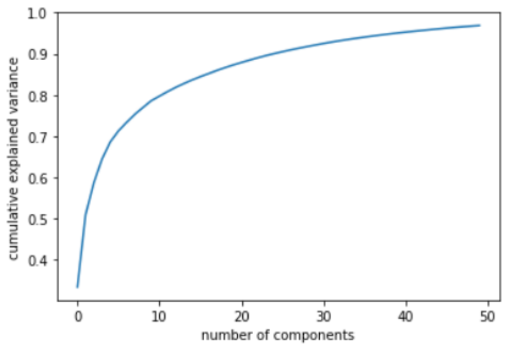


Figure 4: Cumulative explained variance ratio vs Number of components

In order to determine the number of components needed to describe the data, we need to look at the cumulative explained variance. The figure 3 shows the graph of cumulative explained variance ratio against the number of components.

A variance of 95% was set as a threshold to select the number of principle components. As observed from figure 3, **a variance of 95.02% was achieved with 50 principle components.**

2.5 Face Recognition

2.5.1 SVM vs kNN

One of the key considerations regarding the facial recognition component is the classification algorithm. We experimented with both kNN and SVM for the classification task. However, the classification performance of kNN rapidly deteriorated with the increase in number of dimensions. On the other hand, SVM was able to achieve good prediction for new observations despite large number of dimensions. The poor performance of kNN could be the result of providing equal attention to all variables.

Finally, computation and memory resources are important practical considerations for real-time prediction. SVM only needs a small subset of training points (the support vectors) to define the classification rule, making it often **more memory efficient and less computationally demanding** during inference. In contrast, kNN typically requires higher computation and memory resources because it needs to use all input variables and training samples for each new observation to be classified. Therefore, **SVM was opted to be used as the classification algorithm** for the face recognition.

2.5.2 Real-time inference

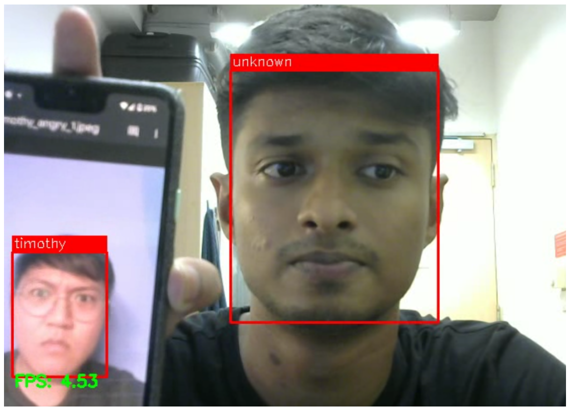


Figure 5: Real-time recognition for multiple faces

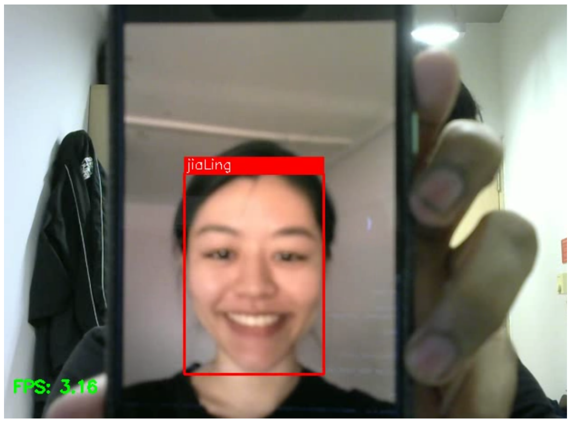


Figure 6: Real-time recognition for single face

OpenCv's Videocapture module was used to capture each frame and was processed in the face recognition system. The system was also able to **detect unknown faces.**

3 Discussion & Analysis

3.1 PCA Reconstruction

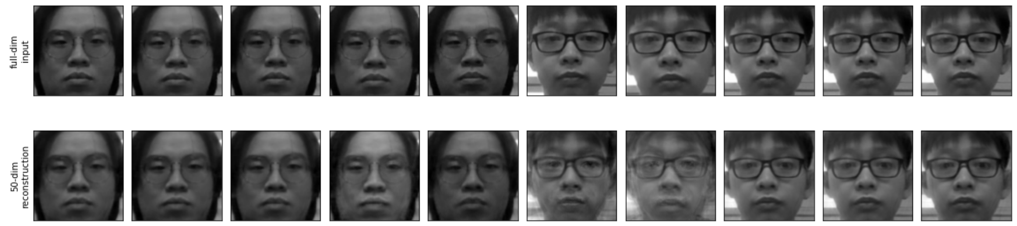


Figure 7: Reconstruction of the face images from 50 principle components

To illustrate the effectiveness of the PCA in feature extraction, we compare the reconstruction of the faces from the chosen principle components. The reconstruction is close to the original face image, therefore we conclude that the 50 principle components is adequate to represent the important facial features.

4 Conclusion

The team managed to accomplish the project focus that was set at the beginning of the project. Based on the output of the PCA Face Recognition System using the dataset created by the team, the team would like to propose some future recommendations to improve the effectiveness and efficiency of the Face Recognition System.

Future teams can expand the dataset further. Since the machine learning algorithm is data-intensive, more data available would help the model's effectiveness. When expanding the dataset, future teams can include images or data that includes lighting. By adding the images, the dataset would be more diverse, which can help the model to be more robust and effective.

Appendices

PCA Implementation

```
class PCA:

    def __init__(self, n_components=None, whiten=False):
        self.n_components = n_components
        self.whiten = bool(whiten)

    def fit(self, X):
        r,c = X.shape
        X = X.astype(np.int32)

        self.mn = mean(X,axis=0)
        X = X - self.mn

        if self.whiten:
            self.stdv = std(X,axis=0)
            X = X/self.stdv

        # Step 1: Calculating covariance matrix
        A = np.array(X)
        C = cov(A.T)

        # Calculating eigen values and eigen vectors
        self.eigval, self.eigvect = eig(C)

        # Step 2: Truncating for dimensionality reduction
        if self.n_components is not None:
            self.eigval = self.eigval[:self.n_components]
            self.eigvect = self.eigvect[:, :self.n_components]

        # Step 3:
        # Sorting the eigen values and vectors in descending order
        desc_ord = np.flip(np.argsort(self.eigval))    # Indices returned are for ascending
    order. Flipping to return indices in descending order.
        self.eigval = self.eigval[desc_ord]
        self.eigvect = self.eigvect[:, desc_ord]

        return self

    # Transforming the original matrix(of features) to the reduced form.
    def transform(self, X):
         X = X - self.mn
         if self.whiten:
             X = X/self.stdv
         return X @ self.eigvect           # Step 4

    def fit_transform(self, X):
        self.fit(X)
        return self.transform(X)

    # Defining the property explained_variance_ratio. This is not a method
    # .explained_variance_ratio
    @property
    def variance_ratio(self):
        return (self.eigval/np.sum(self.eigval))*100   # How much each eigen value
    contributes to the variance in the data.
```