Face Recognition by LCA, PCA, and Fine-tuning of Convolutional Neural Networks of AlexNet and VGG-16

KangKang Li

Abstract

Recognition of human face is a technology growing explodingly in recent years. Eigenface, Fisherface, k-nearest neighbors, support vector machine, and sparse representation classification were implemented with the optimization of vector selection, whitening, and the size of training data. Dropping top 5 eigenvectors significantly improved the accuracy of eigenface with k-nearest neighbors from 42%to 74%, but not for others. Fisherface shows overall higher accuracies than eigenface, and fisherface with sparse representation classification is the optimum combination with an accuracy at 97%. In the second part, this project fine-tunes and evaluate pre-trained CNNs. The performance of AlexNet and VGG-16 before and after fine-tuning by Euclidean distance is discussed in receiver operating characteristic curve. VGG-16 shows better performance than AlexNet, and improvement of both networks was observed with fine-tuning.

1 Introduction

Face recognition starts from the most intuitive way based on the geometric features of a face. This technology relies on algorithms to process and classify digital signals from images or videos. This project helps understanding the ideas and architecture of fundamental algorithms. This approach is limited by the complicate registration of the marker points, even with state of the art algorithms. Comparatively, another family of algorithms treats images and faces as vectors to classify in a n-domentional space, which is described and introduced in the following paragraphs.

2 Methods

2.1 Eigenfaces

The Eigenfaces algorithm[1] transforms high-dimensional image space to a lower-dimensional representation subspace by Principal Component Analysis (PCA), which identifies the axes with maximum variance. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in R^d$.
Compute the mean μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

Compute the the Covariance Matrix S

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T \tag{2}$$

Compute the eigenvalues λ_i and eigenvectors v_i of S

$$Sv_i = \lambda_i v_i, i = 1, 2, \ldots, n \tag{3}$$

Order the eigenvectors descending by their eigenvalue. The k principal components are the eigenvectors corresponding to the k largest eigenvalues.
The k principal components of the observed vector x are then given by:

$$y = W^T(x - \mu) \tag{4}$$

where $W = (v_1, v_2, \ldots, v_k)$. The reconstruction from the PCA basis is given by:

$$x = Wy + \mu \tag{5}$$

The Eigenfaces method then performs face recognition by:
Projecting all training samples into the PCA subspace.
Projecting the query image into the PCA subspace.
Finding the nearest neighbor between the projected training images and the projected query image.

To solve the covariance matrix $S = XX^T$, a transformation is performed to take the eigenvalue decomposition $S = X^T X$ of size NXN instead:

$$X^T X v_i = \lambda_i vi \tag{6}$$

and get the original eigenvectors of $S = XX^T$ with a left multiplication of the data matrix:

$$XX^T(Xv_i) = \lambda_i(Xv_i) \tag{7}$$

The resulting eigenvectors are orthogonal, to get orthonormal eigenvectors they need to be normalized to unit length for further classification[2].

2.2 Fisherface

While the Eigenface using PCA transformation is optimal from a reconstruction standpoint, it doesn't take any class labels into account. Therefore a class-specific projection with a Linear Discriminant Analysis (LDA) was applied to face recognition in [3][4]. The basic idea is to minimize the variance within a class, while maximizing the variance between the classes at the same time. In order to find the combination of features that separates best between classes the LDA maximizes the ratio of between-classes to within-classes scatter. The idea is simple that same classes should cluster tightly together, while different classes are as far away as possible from each other. The algorithm is shown as follow:

Let X be a random vector with samples drawn from c classes:

$$\begin{aligned} X &= \{X_1, X_2, \ldots, X_c\} && (8) \\ X_i &= \{x_1, x_2, \ldots, x_n\} && (9) \end{aligned}$$

The scatter matrices S_B and S_W are calculated as:

$$\begin{aligned} S_B &= \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T && (10) \\ S_W &= \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T && (11) \end{aligned}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{12}$$

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j \tag{13}$$

Fisher's classic algorithm now looks for a projection W, that maximizes the class separability criterion:

$$W_{opt} = arg\, max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{14}$$

Following [3], a solution for this optimization problem is given by solving the General Eigenvalue Problem:

$$\begin{aligned} S_B v_i &= \lambda_i S_w v_i \\ S_W^{-1} S_B v_i &= \lambda_i v_i && (15) \end{aligned}$$

The rank of S_W is at most $(N - c)$, with N samples and c classes. Similar to Eigenface, the reduction in dimention was solved [3] by performing a PCA on the data and projecting the samples into the $(N - c)$-dimensional space.

The optimization problem can be rewritten as:

$$W_{pca} = arg\,max_W |W^T S_T W| \tag{16}$$

$$W_{fld} = arg\,max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|} \tag{17}$$

The transformation matrix W, that projects a sample into the $(c-1)$-dimensional space is then given by:

$$W = W_{fld}^T W_{pca}^T \tag{18}$$

2.3 Support Vector Machine

Support Vector Machines (SVM) were first introduced by Vapnik and Chervonenkis[5]. The core idea is to find the optimal hyperplane to seperate different classes. While there are theoretically infinite hyperplanes to seperate the dataset, an optimum hyperplane is chosen so that the distance to the nearest datapoint of both classes is maximized. The points spanning the hyperplane are the *Support Vectors*[6]. Given a Set of Datapoints $\mathcal{D}$:

$$\mathcal{D} = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1, 1\}\}_{i=1}^n$$

where x_i is a point in p-dimensional vector, and y_i is the corresponding class label
We search for $\omega \in R^n$ and bias b, forming the Hyperplane H:

$$\omega^T x + b = 0$$

that seperates both classes so that:

$$\omega^T x + b = 1, \text{if } y = 1$$
$$\omega^T x + b = -1, \text{if } y = -1$$

The optimization problem that needs to be solved is:

$$\min \frac{1}{2} \omega^T \omega$$

$$\omega^T x + b \geq 1, y = 1$$
$$\omega^T x + b \leq 1, y = -1$$

The kernel trick is used for classifying non-linear datasets. It works by transforming data points into a higher dimensional feature space with a *kernel function*, where the dataset can be seperated again. Commonly used kernel functions are *RBF kernels*:

$$k(x, x') = exp(\frac{\|x - x'\|^2}{\sigma^2})$$

or *polynomial kernels*:

$$k(x, x') = (x \cdot x')^d$$

.

2.4 Sparse Representation Classification

Sparse Representation-based Classification(SRC) is an algorithm represent query images with all the samples in the training set, proposed by Wright[7]. This is equal to solve the linear system of equation y = Ax, where y are query images and A is the matrix columns of training samples. Since linear system of equations is usually underdetermined, the target of SRC is to find sparsest solution by L1-minimization instead of unique solution.

$$\arg\min_{\infty} = \arg\min \|w\|_1 \tag{19}$$

For classification task, the image was reconstructed corresponding to each class of train samples to compute the residual of it between the real query image. Its intuitive idea is to reconstruct with the smallest residual, which is most likely to be the true class of query image. The process can be expressed as:

$$\min_i r_i(y) = \|y - A\gamma_i(\hat{x})\|_2 \tag{20}$$

where γ_i is a function that selects the coefficients within the ith class.

2.5 Fine-tuning using CNN of AlexNet and VGG-16

Neural network is the most efficient and promising tool for image recognition. Various convolutional neutral networks (ConvNet) were develop and showed excellent accuracy for image recognition. Among these networks, AlexNet and VGG-16 are milestones in this area and fundamental models to begin with. This study applied fine-tuning on AlexNet[8] and VGG-16[9] models with LFW dataset.

AlexNet is the first prevalent ConvNet model for image recognition, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton[8]. AlexNet achieved top-1 and top-5 error rates of 37.5% and 17.0% which was the state-of-art result in 2012. It has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax, as shown in Figure 1. Non-saturating neurons and dropout were used to accelerate training and reduce overfitting. In the ILSVRC-2012 competition, AlexNet achieved a winning top-5 test error rate of 15.3%, compared to 26.2% of the second one.

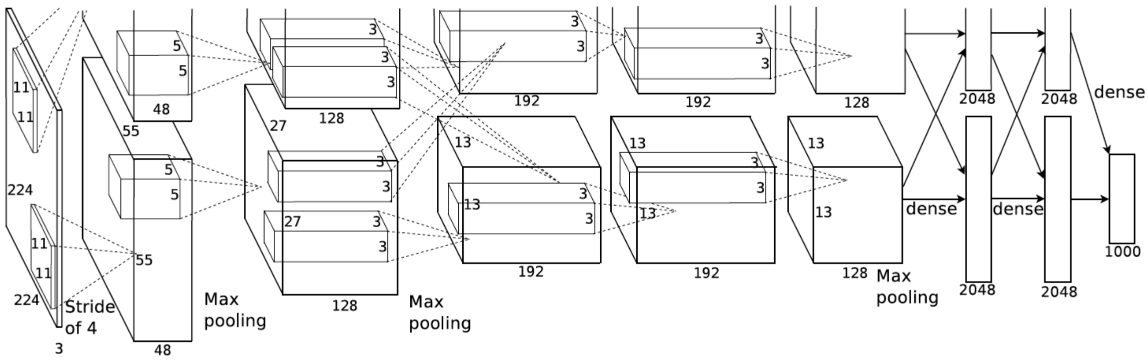

Figure 1: Network structure of AlexNet

VGG is introduced by Ken Chatfield et al. in 2014, and VGG-16 is one model of VGG that has 16 layers[9]. Improvement over AlexNet was made in building of VGG model, by replacing large filters with multiple smaller filters, as shown in Figure 2. In many situations, multiple stacked smaller filters is better than the one with a larger one. The reason may be that multiple non-linear layers increases the depth of the network so that it learn more complex features with lower cost. In ImageNet Challenge 2014, VGG won first and the second places in the localization and classification tracks respectively. VGG also generalize well to other datasets.

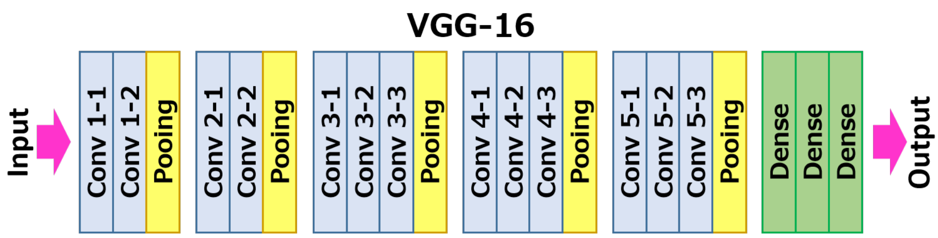

Figure 2: Network structure of VGG

3 Experiment and Discussion

3.1 Implementation of PCA and LDA for image classification

The experiment is performed on a dataset from Yale database which includes around 64 near frontal images under different illuminations of each of 38 individual. In this study, certain number of images from each individual p = (10, 20, 30, 40, 50) was used for training and the rest for testing. Multiple python libraries were used in the experiment including LDA, kNN and SVM from sklearn and SRC from skmultilearn.

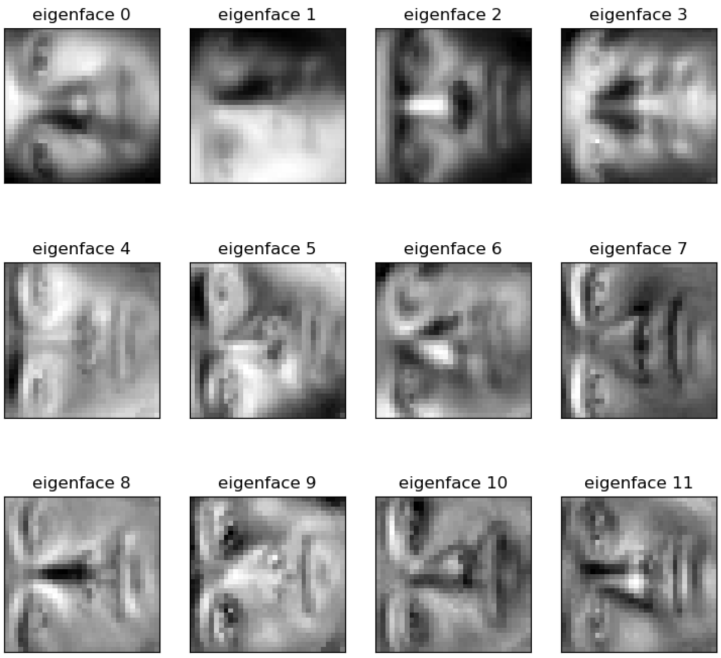

Figure 3: Eigenfaces obtained from YaleB dataset

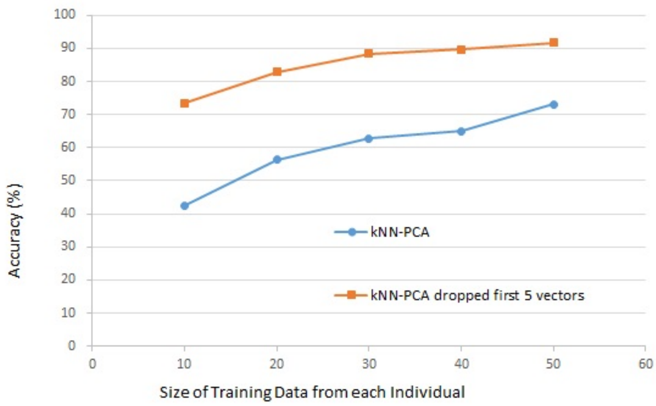

Figure 4: Effect of dropping top eigenvectors

Figure 3 shows eigenfaces obtained from YaleB dataset. Top eigenfaces is related to the illuminations, and may affect the accuracy of face recognition. The accuracy of eigenfaces with and without dropping top eigenfaces were tested and shown in Figure 4. The accuracy after dropping increases by $\sim 20\%$ with kNN as the classifier. This result supports the assumption of illumination. The accuracy increases with the size of training from 73% to 92%.

Anothor key paramter of eigenface is the number of vectors. The number of 100 and 150 were tested and shown in Figure 5. Higher accuracy (90% vs 85% at p=50) is observed at number of vectors is 150, as an effect of classification at higher dimension. This number of 150 is used in the following study.

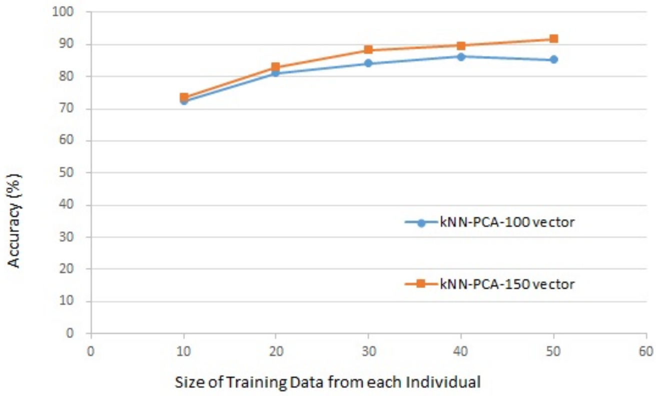

Figure 5: Effect of number of eigenvectors

PCA and LDA were implemented with classifiers of kNN, SVM, and SRC in Figure 6.Unlike PCA, dropping top eigenfaces (eigenvectors) does not affect LDA. Whitening was required in SVM and applied in all tests in Figure 6. LDA shows overall higher accuracy than PCA with each classifier, due to consider images grouply instead of individually as a class. SVM gives higher accuray than kNN, and SRC gives higher accuracy than SMV generally for both PCA and LDA. The highest accuracy was obtained at LDA with SRC.

Figure 6: PCA and LDA test using kNN, SVM, and SRC as classifier

3.2 Fine-tuning using AlexNet and VGG-16

The network models were trained on Tensorflow 1.8.0. ROC curves and images were generated by skliearn and matplotlib, respectively. AlexNet and VGG-16 models, and pretrained weights were obtained from kratzert and Toronto University. The dataset used for training is YaleB.

AlexNet and VGG-16 were implemented and loaded with weights for all layers except the last one. The features were obtained as the second last fc layer before and after training. Learning parameters were used as follow: learning rate at 0.1 and 0.01, epoch: 200 and 20, and batch size of 128. ROC curve was plotted with euclidean distance. Because of the lack of GPU, the training epoch is very limited to merely show the concept of fine tuning.

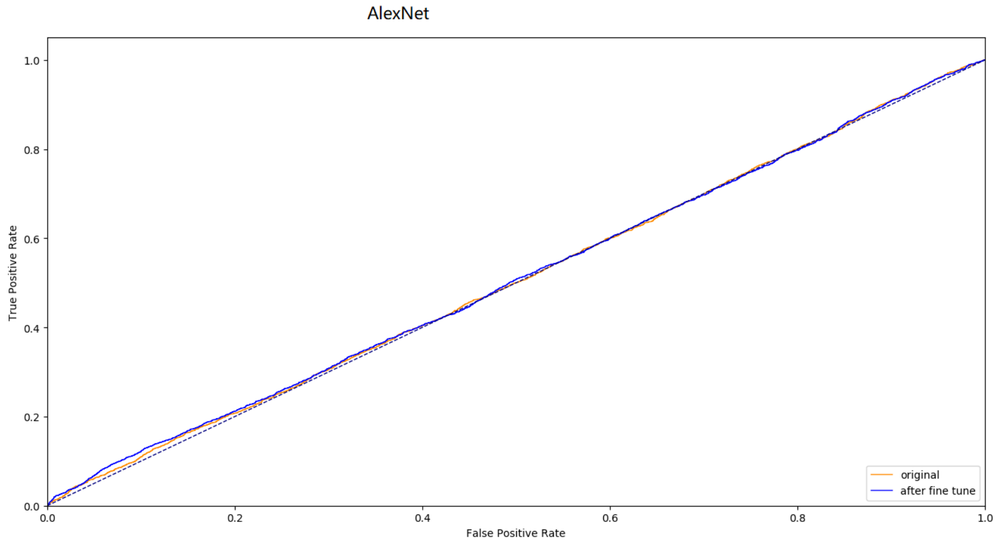

Figure 7: ROC curve of AlexNet before and after fine tuning

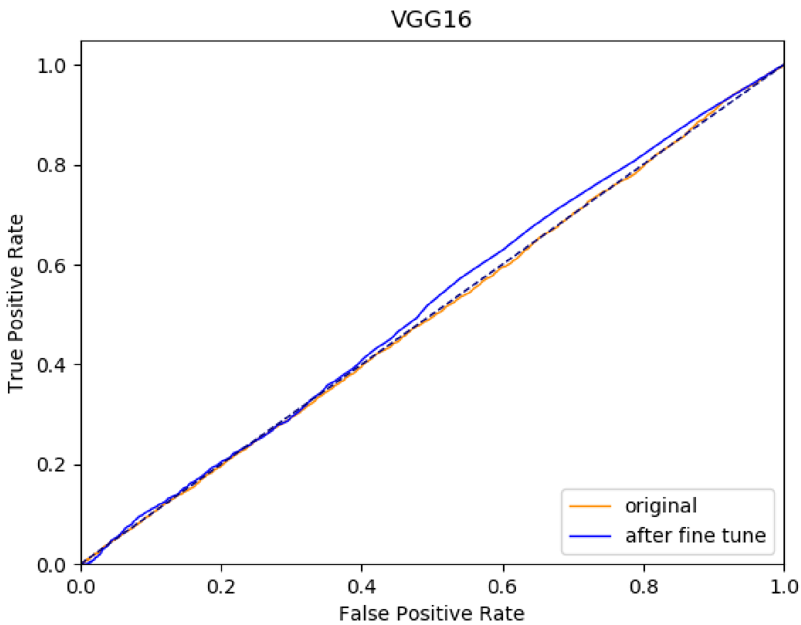

Figure 8: ROC curve of VGG before and after fine tuning

Because of the strong illumination bias of YaleB database, the accuracy is close to 50-50. After fine tuning AlexNet, slight improvement is shown in Figure 3. For VGG-16, the improvement after fine tuning is more evident. The epoch used for VGG is only 20 comparing to 200 for AlexNet due to much longer training time for VGG and the lack of GPU for both networks, the author assume the improvement would be more evident with sufficient training.

4 Conclusions

This study implemented Eigenface (PCA) and fisherface (LDA) with classifiers of kNN, SVM, and SRC on YaleB dataset. The accuracy was improved by dropping top eigenvectors of PCA with kNN and increasing size of training data. In generall, LDA shows higher accuracy than PCA, and SRC shows higher accuracy than kNN and SVM. The highest accuracy at 97% was obtained using LDA and SRC. In the second part, this project fine-tunes and evaluate pre-trained Networks by face recognition. The performance of AlexNet and VGG-16 before and after fine-tuning by Euclidean distance is discussed in receiver operating characteristic curve. VGG-16 shows better performance than AlexNet, and improvement of both networks was observed with fine-tuning.

References

[1] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. 2 edition, November 2001.

[3] Peter N. Belhumeur, Joo Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[4] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

[5] V. N. Vapnik and A. Ya. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, USSR, 1974.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, volume 20, pages 273–297, 1995.

[7] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.