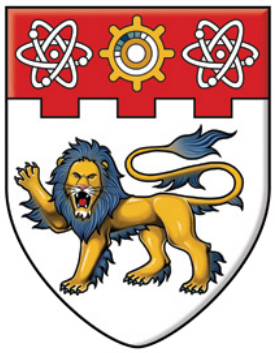


NANYANG TECHNOLOGICAL UNIVERSITY

CZ4041 Machine Learning Project Report

Instructor: Prof. Xu Dong, Prof. Kong Wai-Kin Adams

Group 17

Zhang Danyang

Zhang Xingjia

Wei Xiaoliang

SCHOOL OF COMPUTER ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

In this project, a number of machine learning algorithms are explored using python and related libraries. These include PCA, kernel PCA, Fisherface, NDA and LGBPHS. There are two databases adopted in this project, namely, the ORL Database of Faces and the Yale Face Database.

This report gives a brief description of the dataset, algorithms, experiments and results analysis. In addition, the performance of combinations of these algorithms are also explored and testified in the experiments section.

2 DATASET

Originally we used ORL database, but it is very easy to achieve high precision even using PCA, thus we use Yale A database in this report instead.

The first data set used in this project is "The ORL Database of Faces" [1].

It contains face images of 40 different subjects. For each subject, there are 10 images that have distinct variations in lighting, facial expressions (open/closed eyes, smiling/not smiling) and configurations (glasses/no glasses). Moreover, the ORL database is very easy to achieve high results (e.g. PCA 98%),

The second data set used in this project is the Yale Face Database [2].

The Yale Face database consists of 165 grayscale GIF images of 15 individuals. Each subject (individual) has 11 images that have distinct variations in facial expression or configuration. These include center-light, with glasses, happy, left-light, without glasses, normal, right-light, sad, sleepy, surprised and wink.

However, the images of Yale Face dataset are not aligned Thus there is a need to pre-process the images to align them and crop out the face area. Fortunately, the coordinates of both eyes of the images are available on the internet. The images are cropped by considering the eye coordinates with 100x300 in resolution. The processed images are stored in separate folders for each person. Thus there are a total of 15 folders and 11 images in each folder.

Both databases have differences in facial expression and configurations. These variations can increase the difficulty of face recognition, thus it can better measure an algorithm's performance when there is noise in the image.

3 METHODOLOGY

This section describes the methods used in this project.

3.1 K-FOLD

K-fold cross validation is a technique that is used to assess the performance of machine learning algorithms due to finite amount of data used in this project. It divides the data for training and testing separately into k partitions. One partition will be used for testing while the k-1 partitions will be used for training. Therefore, all data will eventually be used for both training and testing purposes. Once the k-fold cross validation is complete, the results will be averaged to give a general evaluation for the algorithm.

3.2 PRECISIONS OF MULTI-CLASS CLASSIFIER

The precision is evaluated by plain multi-class kNN classifier. A test data is assigned to a class label according kNN, and the precision is calculated as followed:

$$precision = \frac{\#correct\ assignments}{\#correct\ assignments + \#incorrect\ assignments}$$

3.3 BINARY APPROACH FOR COMPUTING MULTI-CLASS CLASSIFIER ROC

In this project, an ROC curve is plotted for each person. Then the multi-class ROC is calculated from these ROC curves of each individual. Each time, the classification problem is considered as a binary problem. Then the mean value of (TPR, FPR) for all the classification problems is computed given an arbitrary threshold value.

This means that for the i-th classification problem in the j-th fold, a function τ_{ij} that maps a threshold value to (TPR, FPR) is calculated:

$$\tau_{ij}: threshold \rightarrow (TPR, FPR)$$

After that, the results of different values of i and j are averaged. The resulted (TPR, FPR) is then used to plot the multi-class ROC curve.

3.4 K-NEAREST NEIGHBORS (KNN)

K-Nearest Neighbours (kNN) is a classification method that predicts unlabeled data according to their similarity against the training data.

The kNN algorithm only requires the following:

1. An integer k
2. A set of labeled examples
3. A metric to measure distance among neighbors

In this project, discussion of possible values of integer k can be found in section Experiment Setting, Result and Analysis experiment 1. The face datasets used in this project is also labeled. Moreover, Euclidean distance serves as the metric to measure the distance among neighbors. Euclidean distance can be obtained by:

$$E(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

where m is the number of features, x and y are data samples.

3.5 Principal Component Analysis (PCA) & Kernel PCA

3.6 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that preserves as much variance in the high dimensional space as possible. It projects the whole set of data onto a different subspace [3].

Suppose we have the following **d** x **N** matrix:

$$X = [x_1, x_2, \ldots, x_N]$$

where there are **N** input data with **d** dimensions. The covariance matrix can be defined as follows:

$$C = \frac{1}{N} X X^T$$

Then the eigenvectors $U$and eigenvalues Λ can be computed using eigendecomposition:

$$CU = U\Lambda \Rightarrow C = U\Lambda U^T = \sum_a \lambda_a u_a u_a^T$$

where C, U and $\Lambda$are all square matrix. $U = [u_1, u_2, \ldots, u_d]$ and $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_d)$.

In order to perform dimensionality reduction, the eigenvalues are sorted in decreasing order and the first **k** eigenvectors, $U_k = [u_1, u_2, \ldots, u_k]$, are selected. Then the **k** dimensional feature subspace **Y** can be obtained by:

$$Y = U^T X$$

As such, the d dimensional $x_i$is projected to the k dimensional $y_i = U_k^T x_i$.

In the case that **d** is large and the C is of dimension **d**x**d,** the eigenvectors and eigenvalues is computationally expensive to compute, thus we use PCA transpose trick to speed up the calculations for U and Λ.

$$C^* = \frac{1}{N} X^T X$$

$$C^* U^* = U^* \Lambda^*$$

since C* is of dimension NxN that is significant smaller than dxd, eigenvectors $U^*$and eigenvalues Λ^*can be computed with less time. The following equations drives U and Λ:

$$XC^* U^* = XU^* \Lambda^*$$

$$X \frac{1}{N} X^T X U^* = XU^* \Lambda^*$$

$$C(XU^*) = XU^* \Lambda^*$$

Thus, $U = XU^*$ and $\Lambda = \Lambda^*$.

3.7 KERNEL PCA

PCA works well if the input data is linearly separable. However, if the given data is linearly inseparable, a nonlinear technique is needed as a result. Kernel PCA is a nonlinear form of PCA. It applies kernel functions to compute principal components in high dimensional feature spaces [4].

Suppose that there is a **d** dimensional feature space that is to be transformed to an **M** dimensional feature space where $M \gg d$ via a transformation function $\Phi(x)$. Then for each point x_i, $\Phi(x_i)$ will be the projected point. Although PCA can be performed, the process could be computationally expensive and inefficient. However, with the use of kernel methods, the computation process could be simplified.

Assume that the projected new features has zero mean value (i.e. zero-centered data):

$$\frac{1}{N} \sum_{i=1}^{N} \Phi(x_i) = 0$$

Then the **M** x **M** covariance matrix can be calculated from:

$$C = \frac{1}{N} \sum_{i=1}^{N} \Phi(x_i) \Phi(x_i)^T$$

If the kernel function is defined by:

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$$

and the both sides of the eigendecomposition equation is multiplied by $\Phi(x_l)^T$:

$$\frac{1}{N} \sum_{i=1}^{N} K(x_l, x_i) \sum_{j=1}^{N} a_{kj} K(x_i, x_j) = \lambda_k \sum_{i=1}^{N} a_{ki} K(x_l, x_i).$$

Then by using the matrix notation:

$$K^2 a_k = \lambda_k N K a_k$$

where K is an M x M kernel matrix, $K_{i,j} = K(x_i, x_j)$ and $a_k$is the N dimensional column vector of a_{ki}:

$$a_k = [a_{k1}, a_{k2}, \ldots, a_{kN}]^{\mathrm{T}}.$$

The $a_k$value can be solved by:

$$K a_k = \lambda_k N a_k$$

Then the kernel principal component can be computed by:

$$y_k(x) = \Phi(x)^T u_k = \sum_{i=1}^{N} a_{ki} K(x, x_i).$$

As mentioned earlier in the report, it is extremely difficult and inefficient to compute $\Phi(x_i)$directly. An alternative is to apply kernel methods which allow us to compute the kernel matrix from the training data set $\{x_i\}$directly. In this way, the computation power becomes cheaper. This is called the "kernel trick". Some of the common kernel methods include the following [5]:

Polynomial kernel: $k(x, y) = (\gamma x^T y + c_0)^d$

Sigmoid kernel: $k(x, y) = tanh(\gamma x^T y + \mathrm{c}_0)$

Cosine kernel: $k(x, y) = \frac{xy^T}{||x|| \cdot ||y||}$

3.8 FISHERFACE

Fisherfaces is a popular method in faces recognition, which makes use of only secondary order statistics. It utilises PCA and linear discriminant analysis (LDA) as a chain operator. Unlike PCA, LDA is class-specific. LDA aims to maximize the between-class scatter and maximize the within-class scatter. These two scatters are defined as:

$$S_w = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T$$

$$S_b = \sum_{j=1}^{C} (\mu_j - \mu)(\mu_j - \mu)^T$$

Where μ_j the mean of class j and μ is the mean all class means. LDA finds the vector V so that $\frac{W^T S_b W}{W^T S_w W}$ is maximized. PCA is applied on the images to reduce the number of dimensions, as explained in the last section. This is because in face recognition, one difficulty is that the within-class scatter matrix is singular, as the rank of $S_w$is at most N-c, where N is the number of images in total, and N is much smaller than number of features each image [6].

Denoting the W obtained in LDA step as W_{LDA} and W obtained in PCA as W_{PCA}, the overall weight of Fisherface is:

$$W_{opt}^T = W_{LDA}^T W_{PCA}^T$$

3.9 Non-Parametric Discriminant Analysis (NDA)

While Fisherface method lies on the assumption that all classes share the Gaussian distribution with the same covariance matrix. Also, the upper bound of the LDA's number of feature is c - 1, same as the rank of the between-class matrix. It is blind beyond second order statistics. The Non-Parametric Discriminant Analysis, on the other hand, uses the data points directly. Specifically, the between class scatter matrix is defined as:

$$S_b = \frac{1}{N} w_n \sum_{n=1}^{N} (\Delta_n^E)(\Delta_n^E)^T$$

where $\Delta_n^E$is the extra-class difference $\Delta_n^E = x - x^E$ for the data sample n, and , $x^E$is the extra-class nearest neighbour.

Assuming x belongs to class C_k, x^E is defined as:

$$x^E = \{x' \in \overline{C_k}, ||x' - x|| \leq ||z - x||, z \in \overline{C_k},\}$$

Similarly, the intra-class cluster can be redefined with x^I, where:

$$x^I = \{x' \in C_k, \ ||x' - x|| \leq ||z - x||, z \in C_k\}$$

w_n is a weight factor to emphasize the points closer to edge between classes defined as:

$$w_n = \frac{min\{||\Delta_n^E||, ||\Delta_n^I||\}}{||\Delta_n^E|| + ||\Delta_n^I||}$$

w_n reaches 0.5 at the class boundaries and gradually reduce to 0 when moving towards the inner points of classes.

This definition can be extended to a K-NN case where x^E, x^I are the mean of the K-nearest neighbours. When K extends to the total available samples in a class, NDA becomes the same as LDA discussed above.

3.10 GABOR FILTERS

Gabor filter is the multiplication of Gaussian Kernel and Fourier Kernel in image convolution.

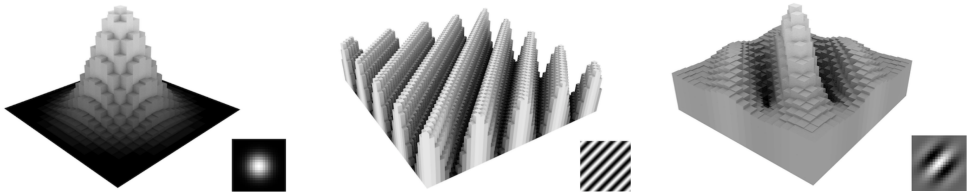


Figure 1 Gaussian Kernel, Fourier Kernel, Gabor Kernel

Gabor filter is defined in the complex form as:

$$\Psi_{u,v}(z) = \frac{||k_{u,v}||^2}{\sigma^2} exp(-||k_{u,v}||^2||z||^2/2\sigma^2)[exp(ik_{u,v}z) - exp(-\sigma^2/2)]$$

$$k_{u,v} = k_v e^{i\phi_u}$$

In the equation, u defines the orientations and $v$defines the scales of Gabor Filter, $k_{u,v}$is wave vector. z represents the position from the origin, $z = (x, y)$. Figure 1 below shows the 40 Gabor filters.

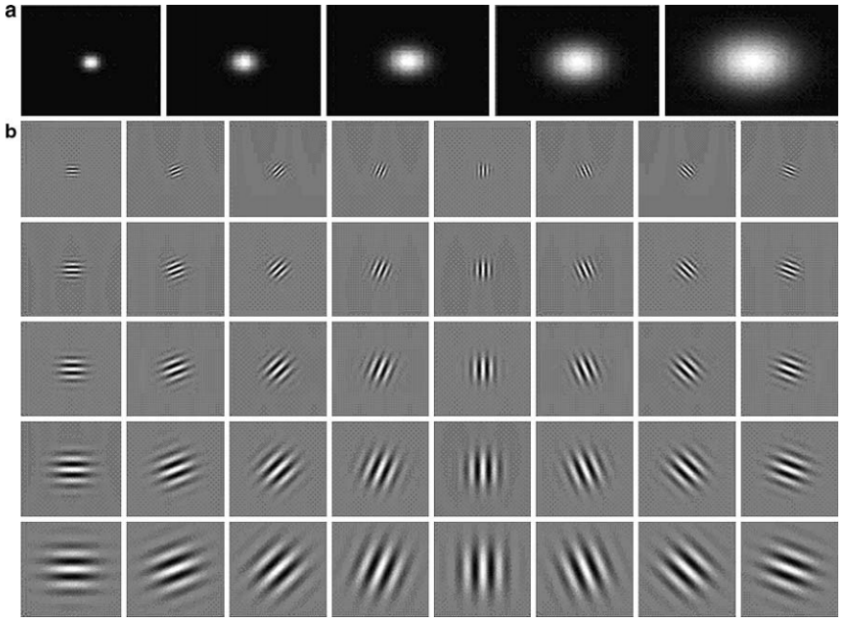


Figure 2: The set of 40 Gabor filters. A, the magnitude at five scales. B, the real parts at 5 scales and 8 orientations

In image processing, an image is convolved with a Gabor filter. The convolution in this project is defined as follow:

$$G_{\Psi f}(x, y, u, v) = f(x, y) * \Psi_{u,v}(z)$$

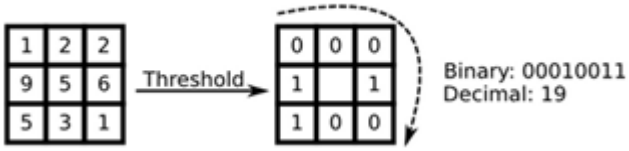
In the equation, $*$ is the convolution operator.

$$f(x,y) * \Psi_{u,v}(z) = \iint f(x - z_x, y - z_y)\Psi_{u,v}(z)\, dz_x\, dz_y$$

In image convolution, we take a weighted average of all the neighbor pixels and use it to update the center pixel of an output image. Every pixel's value was influenced by its neighbors. The set of weights we choose - or the kernel, in this case, is the Gabor filter.

3.11 Local Binary Pattern (LBP) & Circular LBP

Original LBP extracts the local features according to the following equations:



$$S(f_p - f_c) = 1\ if\ f_p \geq f_c\ otherwise\ 0$$

$$LBP = \sum_{p=0}^{7} s(f_p - f_c) \cdot 2^p$$

Circular LBP extends the original LBP, extracting the local features by considering the points on the circle centered at point *c*with radius R.

In circular LBP, the f_p in the original LBP equation becomes the feature of point p at coordinates (x_p, y_p), where:

$$x_p = x_c + Rcos(\frac{2\pi p}{P})$$

$$y_p = y_c - Rsin(\frac{2\pi p}{P})$$

In the equation, P is the total number of neighbors chosen. If the point $p$at (x_p, y_p)does not correspond a pixel in the image, the point is going to be extrapolated from neighboring pixels.

LBP operator is robust against monotonic gray scale transformations. The following figure demonstrates the robustness of LBP of the same subject (Yale A subject 01) under different gray scale conditions:

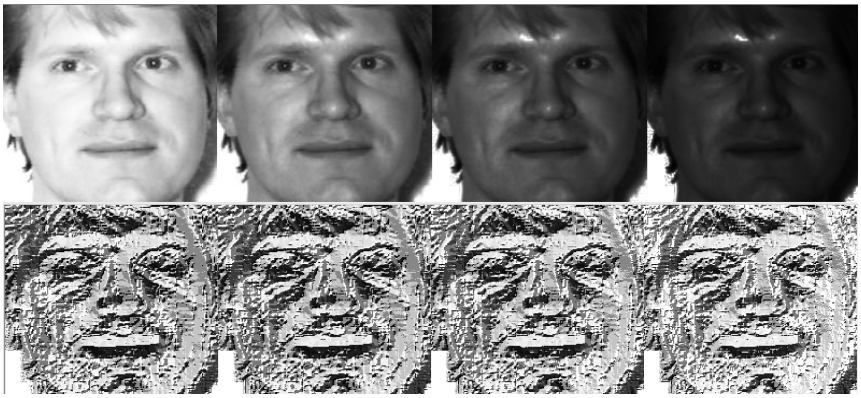


Figure 3: Yale Faces

3.12 Local Gabor Binary Pattern Histogram Sequence (LGBPHS)

Local Gabor Binary Pattern Histogram Sequence (LGBPHS) [7] face recognition approach proposed by Zhang W. et al. The general flow of the algorithm is denoted as Figure 3 below.

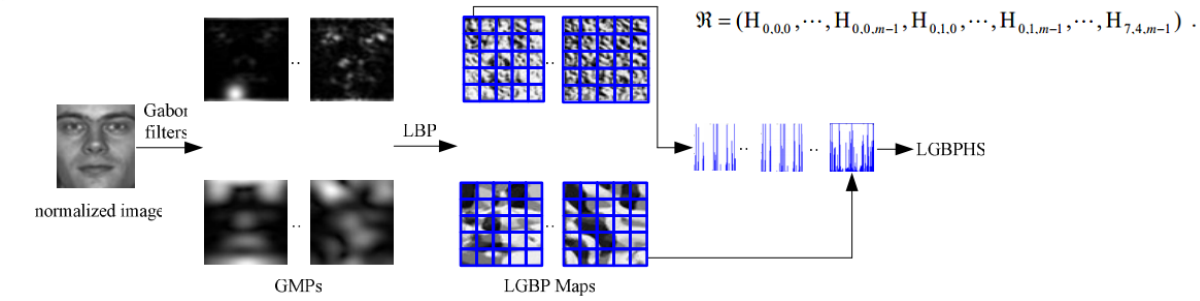


Figure 4: General flow of LGBPHS

In this algorithm, a face image is modeled as a histogram sequence by 4 steps:

1. The normalized input face image is convolved with multi-scale and multi-orientation Gabor filters. Subsequently, we obtain multiple Gabor Magnitude Pictures (GMPs), with each GMP corresponds to one pair of scale and orientation.

2. Each GMP is converted to a Local Gabor Binary Pattern (LGBP) Map by applying LBP (or circular LBP) on it.

3. Each LGBP Map is further divided into multiple non-overlapping rectangular regions with specific size, and a histogram is constructed from each of the region.

4. The histograms from all regions and from all LGBP Maps are contacted to construct the final histogram sequence (LGBPHS) as the model of the input face.

Then the modeled histogram sequence (LGBPHS) of the collection of images will be fed into kNN classifier for the face recognition task. Instead of using Euclidean distance in kNN, histogram intersection is used as the similarity metric:

$$sim(H^1, H^2) = \sum_{i=1}^{L} min({h_i}^1, {h_i}^2)$$

$$sim(\mathrm{R}^1, \mathrm{R}^2) = \sum_{u} \sum_{v} \sum_{r} sim({H_{u,v,r}}^1, {H_{u,v,r}}^2)$$

In the equations, L is the total number of gray scales. R is the final concatenated histogram sequence, $H$is a histogram from a region of a LGBP Map, $u$is for the orientations, $v$is for the scales, and $r$is for the regions.

3.13 Ensemble LBP Fisherface

In this algorithm, as shown in Figure 5, we propose an ensemble learner for farce recognition by manipulating the input feature vector (i.e. the input faces):

1. An input face is pre-processed using circular LBP with certain radius r.
2. The Fisherface constructs the model from the collection of the LBP-processed images.
3. We repeat step 1 and step 2 multiple times with a different radius r; thus we obtain multiple Fisherface models from different representations of the input vector.
4. Multiple kNN classifiers make predictions corresponding to each of the Fisherface model.
5. The ensemble learner combines the kNN classifiers' predictions by majority voting.

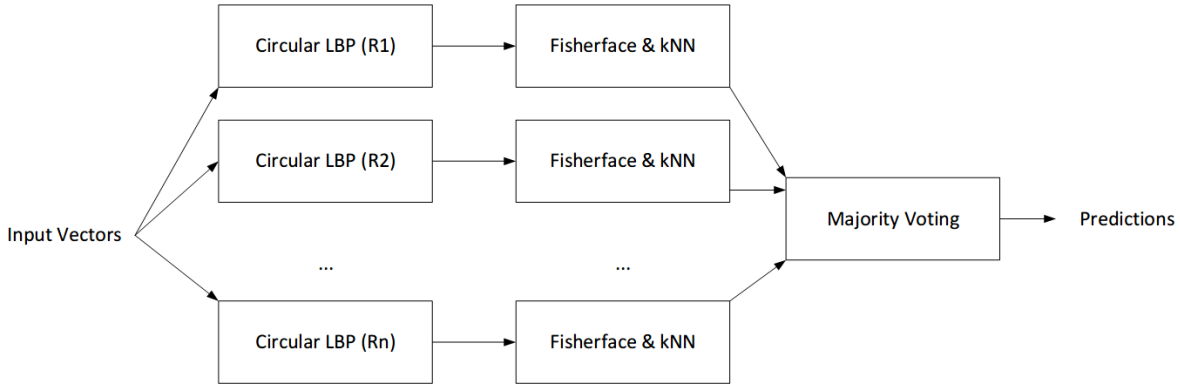


Figure 5: Ensemble LBP Fisherface

4 EXPERIMENT SETTING, RESULT & ANALYSIS

In this project, the relationship between fold size of number of folds is defined as follow.

$$test\ data\ size\ = \frac{number\ of\ data\ per\ individual}{number\ of\ folds}$$

Fold size is the number of test samples per individual in each k-fold iteration, and the remaining samples per individual is the training samples. For example, if the Yale Face Database is used, the number of images individual is 11. When the number of folds is set to 11, the test data size is 11/11 = 1. When the number of folds is set to 2, since the total number of data per person is not divisible by fold number, the test data size is taking the floor value which is 5.

4.1 EXPERIMENT 1: VARYING K FOR KNN

Objective. This experiment aims to find out how various values of k impacts the performance of kNN. It also finds the best value of k used for the following experiments.

Settings. The input data was first fed into PCA to perform dimensionality reduction. The reduced data was then passed to kNN for classification. The number of components used for PCA was kept as 50. k is tested from 1 to 40 with step of 1.

Results.

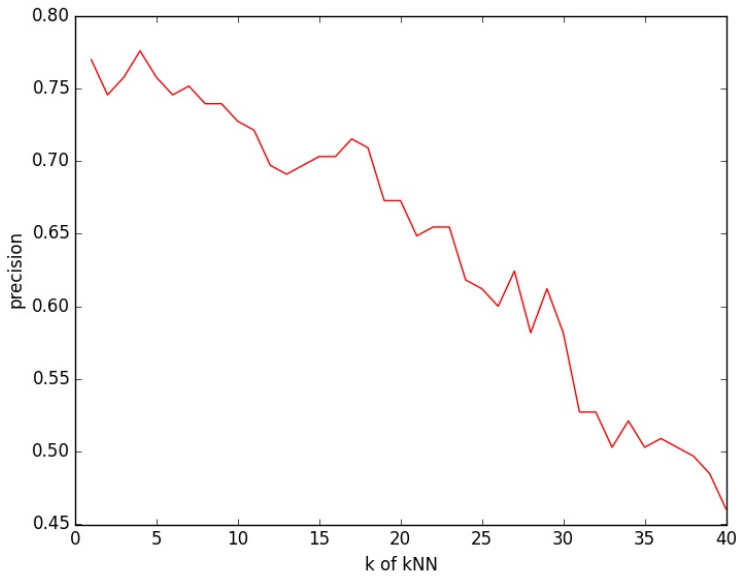


Figure 6: Classification precision for different k of kNN

Analysis. As shown in Figure 6, the precision of kNN generally decreases as values of k increase. This means that the smaller the value of k is, the higher the precision will be. Thus a smaller value of k is to be used for the following experiments because it helps kNN to achieve higher precision. The peak value of precision takes place when k = 4 and the second highest precision happens when k = 1. As the general observation suggests, smaller k value could lead

to higher precision. The peak when k = 4 might best be considered as an outlier as it does not always guarantee to provide a best result. Thus k = 1 will be used instead.

We found that increasing value of k does not help increase the precision. This may be because the data, after dimensionality reduction by performed by PCA, does not form distinct clusters in the Euclidean space, and predicting testing data correctly mainly depends on finding an image of the same class that is very similar to the test image.

4.2 Experiment 2: Comparing PCA and Identity

Objective. This experiment compares the performance of PCA and identity (doing nothing) with a 1NN classifier.

Settings. In this experiment, 1NN was performed on unprocessed input data and data processed with PCA (*num_components* = 50). The experiment is carried out with an 11 fold cross-validation.

Results.

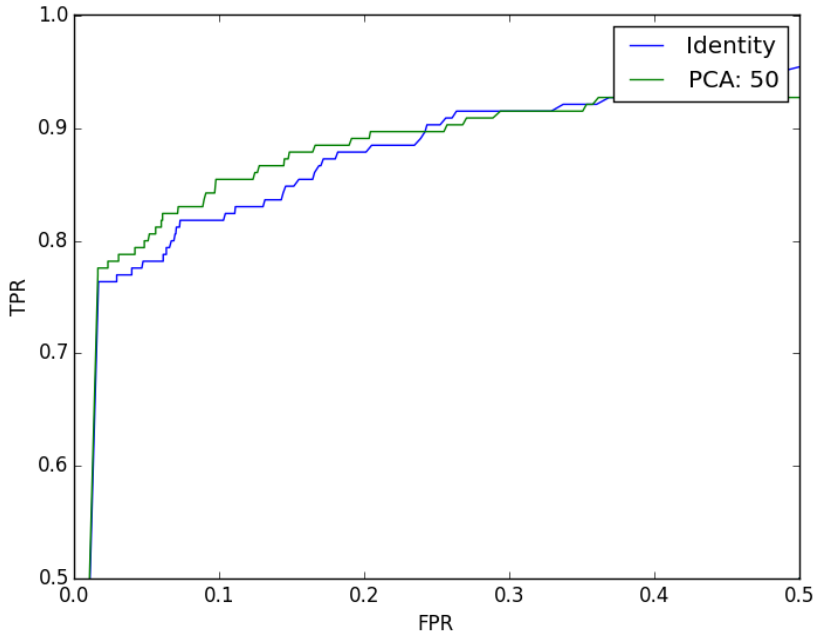


Figure 7: Result of PCA and Identity

As shown in Figure 7, PCA performs gives better performance on regions with lower FPR. PCA also reduces the processing time as it discards many unnecessary features.

4.3 EXPERIMENT 3: VARYING NUMBER OF FOLDS FOR PCA

Objective. This experiment aims to use PCA with 1NN classifier to compare the performance when varying the number of training data and test data by varying the number of folds in cross-validation.

Setting. The range of number of folds is 2 to 11.

Results.

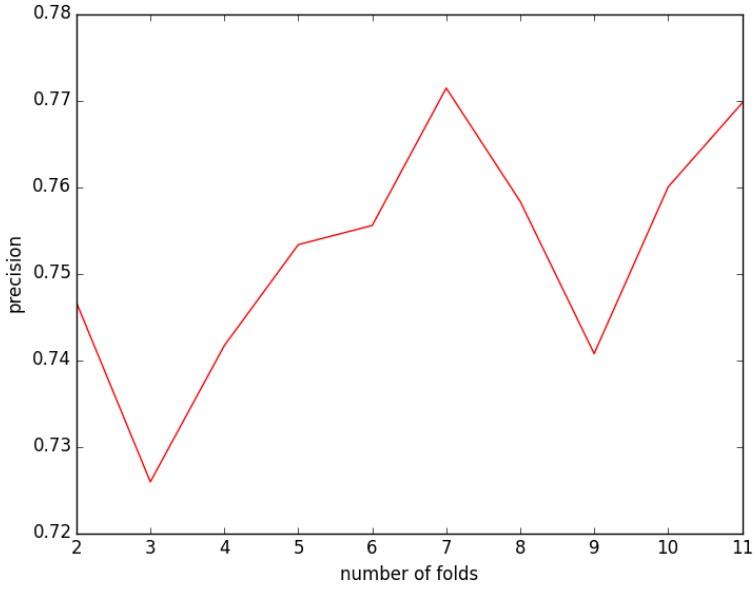


Figure 8: Result of number of folds of PCA

Analysis. As shown in Figure 8, as the number of folds increases, the precision of PCA generally increases. The fluctuation may be caused by the specificity of data as the number of instance per class is limited in Yale A dataset. However, we could see the trend that the more the fold is, the higher the precision. This is because increasing the number of fold essentially decreases the fold size and increase the training data for the model.

4.4 EXPERIMENT 4: KERNEL PCA - VARYING DEGREE OF POLYNOMIAL KERNEL

Objective. In this experiment, its aim is to find out how change in order of degree of polynomial kernel can affect the performance of the kernel PCA.

Settings. The order of degree d's range was set from 1 to 5 while the number of components was kept at 50 throughout this experiment.

Results.

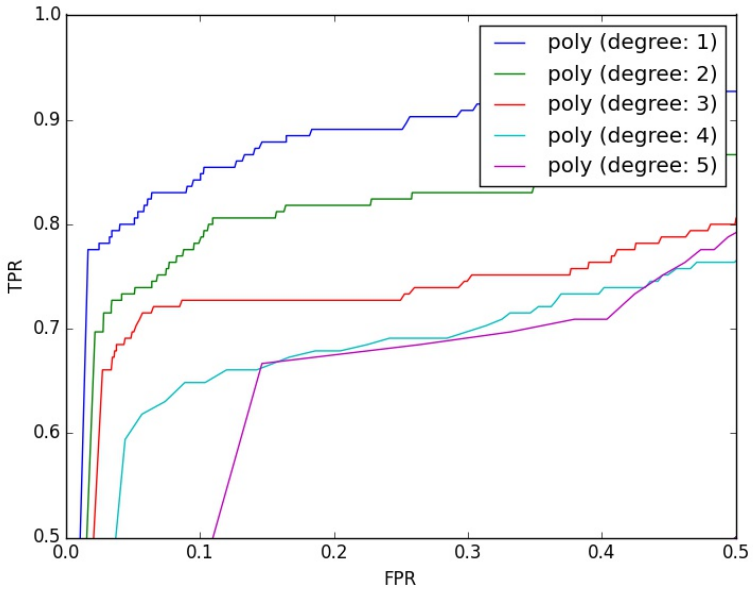


Figure 9: Result of poly kernel with various degree values

Analysis. As shown in Figure 9, the order of degree that achieved the best performance is 1. The performance gradually decreases as the order of degree increases. However, when degree is 1, it is the same as the linear PCA. This may be because the Yale Face database used here does not require to use higher order to be separated.

4.5 EXPERIMENT 5: KERNEL PCA - COMPARISON OF DIFFERENT KERNELS

Objective. This experiment aims to compare the performances of kernel PCA using different kernel methods.

Settings. The degree of polynomial is set to 1 while the gamma value of sigmoid is left as default.

Results.

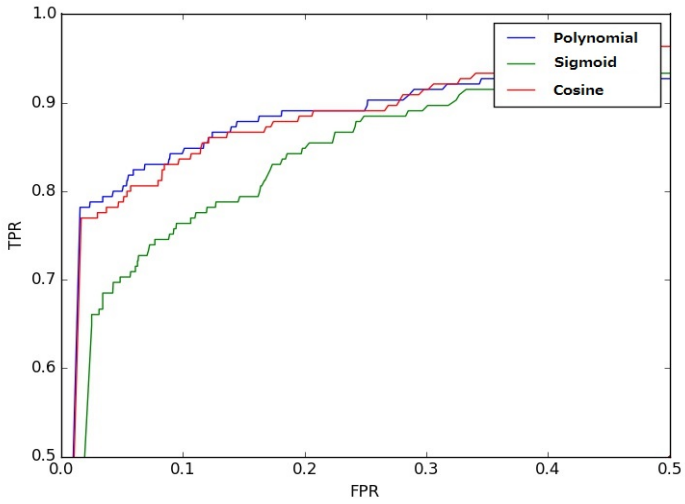


Figure 10: Result of different kernel methods

Analysis. As shown in Figure 10, it can be found that the performance of poly and cosine kernel methods are similar while performance of sigmoid method is lower than the previous two. This may be because the sigmoid kernel does not work well on the Yale Face database.

4.6 EXPERIMENT 6: FISHERFACE - VARYING NUMBER OF PCA COMPONENTS

Objective. In experimenting with Fisherface, we want to evaluate the effect of number of PCA component on the performance of the method.

Settings. In Fisherface, the upper bound for component number is $c - 1$where c is the number of classes, in the case of Yale dataset, 15, a range for 10-14 is used.

Results. As expected, Fisherface's performance grow between with larger number of PCA components. Running multiple experiments shows that the performance stabilizes after 10 and may fluctuate by the specificity of data. In this particular set of experiments, Fisherface exhibits best performance when number of components is at 11 components. The highest average precision is 93.94%.

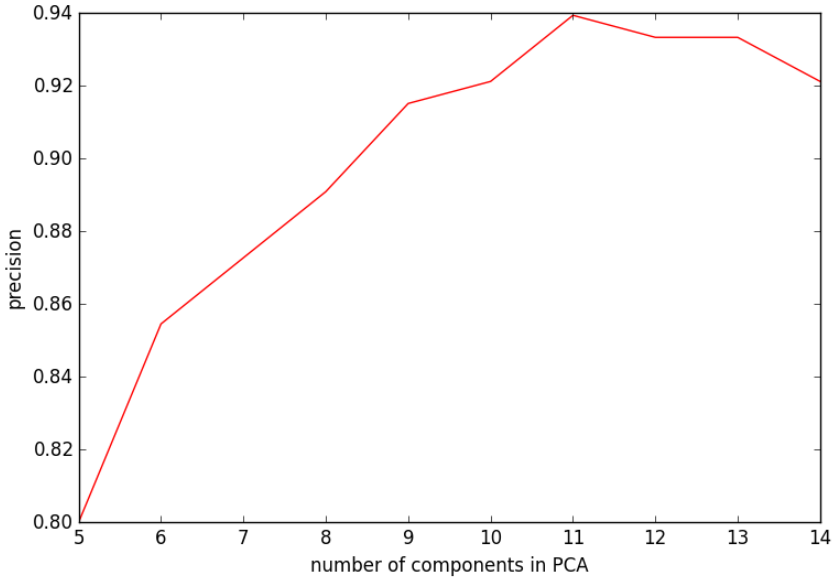


Figure 11: Result of number of components of PCA

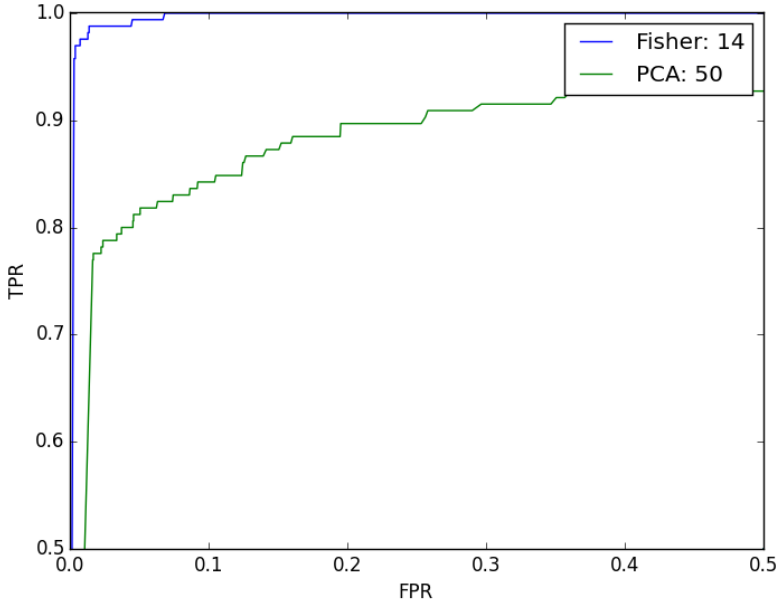


Figure 12: Results of Fisherfaces and PCA

As shown in Figure 12, using 14 PCA components, the ROC of Fisherface is as follows. Here, PCA (with 50 components) is used as a baseline. We can see that Fisherface outperforms PCA.

The average precision of Fisherface and PCA are listed in the table below.

	Fisherface	PCA
Average Precision	93.94%	76.97%

Analysis. We can see that the Fisherface performs better on the dataset than PCA along. If we compare the 14 extracted features with that of PCA (Figure 13), it can be seen that Fisherface (Figure 14) seems to extract the features (regions of a face) that differentiate the persons best from each other. These feature are not influenced by lighting as much as PCA features [8].

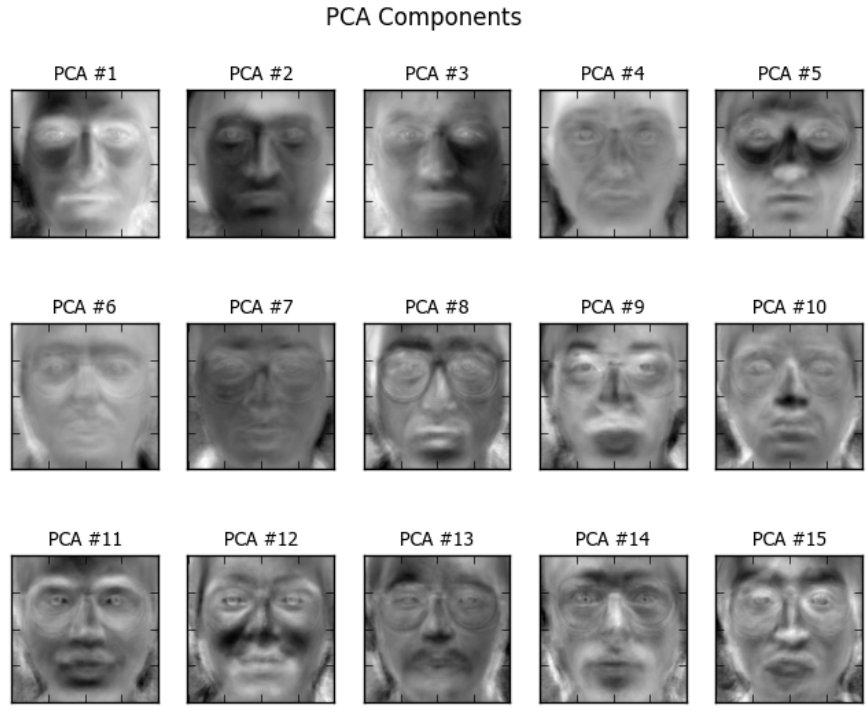


Figure 13: PCA components

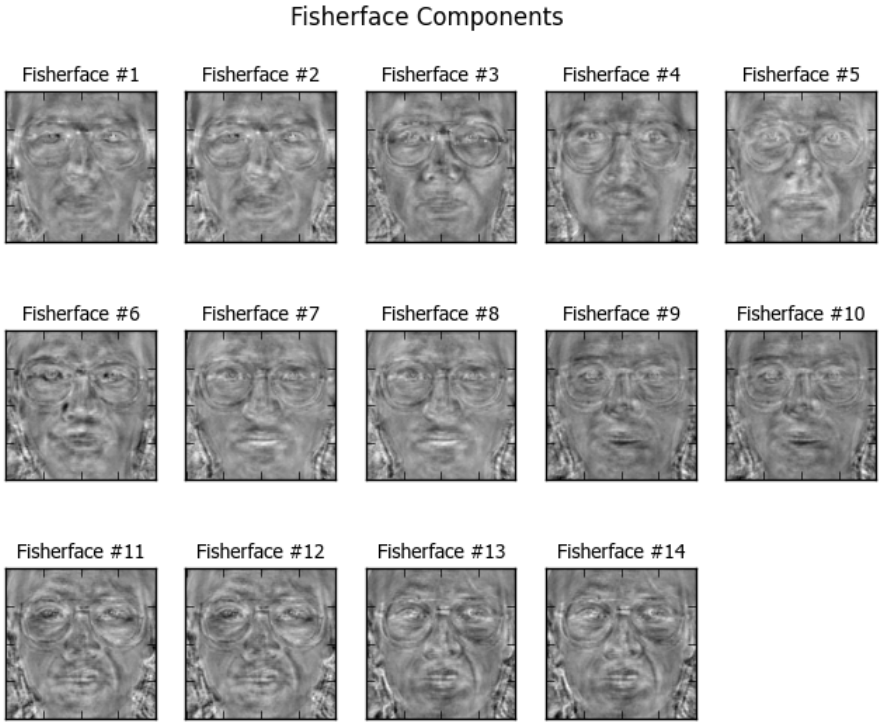


Figure 14: Fisherfaces components

4.7 EXPERIMENT 7: NDA - VARYING PCA COMPONENTS

Objective. The objective for this experiment is to investigate the impact of number of PCA components (*num_components*) on the performance of NDA. An advantage of NDA of LDA is that the number of PCA components does not have an upper bound.

Settings. In this experiment, we changed the LDA part of Fisherface to NDA. We investigated the number of PCA components from 14 to 56 with a step of 7. Around the maximum precision 42 to 50, a step of 2 is used. In this experiment, we use K=1 for K-nearest neighbor in NDA.

Results.

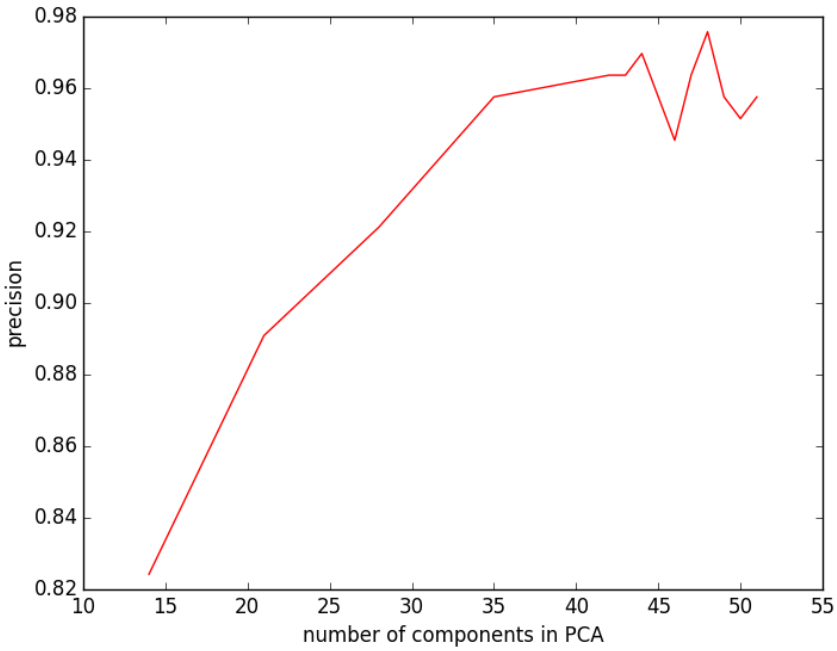


Figure 15: Result of number of components of PCA

While NDA does not perform as good as LDA under same *num_components*, its performance goe up with *num_components*. Average precision of NDA reaches highest, 97.58%, when *num_components* reaches 48, and stays relatively stable with flatuations after that. In general, the performance stables after num_components reaches 40, and the fluctuation may be caused by the specificity of data.

Comparing the ROC of NDA and Fisherfaces in Figure 16, we can see that NDA over performs Fisherface with a narrow edge.

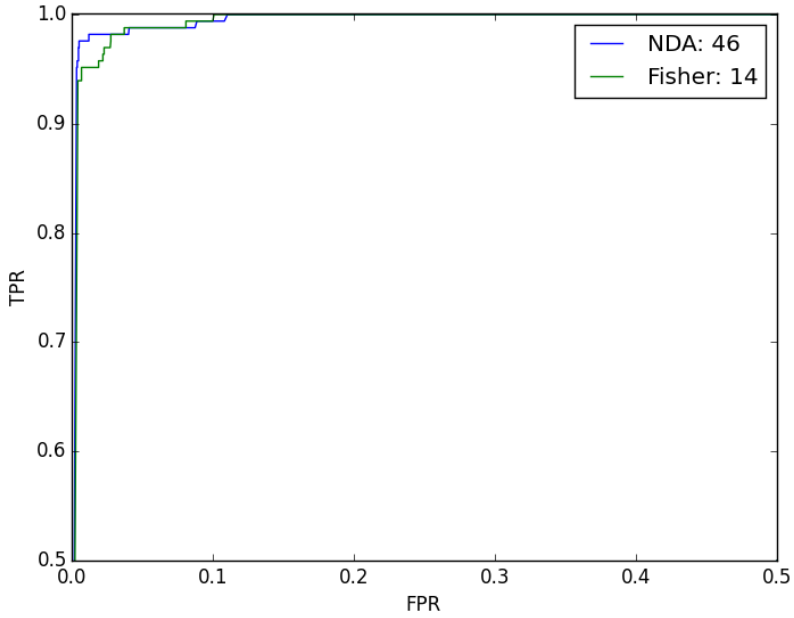


Figure 16: Result of NDA and Fisherfaces

The average precision of NDA and Fisherface are listed in the table below.

	NDA	Fisherface
Average Precision	97.58%	93.94%

Analysis. This experiment shows that the NDA reaches slightly better performance than LDA, but overall performance is similar. This makes sense as NDA is essentially a non-parametric generalization of LDA. In this dataset, the advantage of NDA is that it does not have the limit in number of PCA components and achieves a better precision when PCA component number is much larger than the number of classes.

4.8 EXPERIMENT 8: LGBPHS VS. LBP HISTOGRAM

Objective. The objective of this experiment is to evaluate the performances of LGBPHS (with Gabor Filters) and LBP Histogram (without Gabor Filters).

Settings. The Gabor Filters of LGBPHS is set to **4** different orientations with **2** different scales. The radius of extended LBP is set to **6** with **8** neighbors.

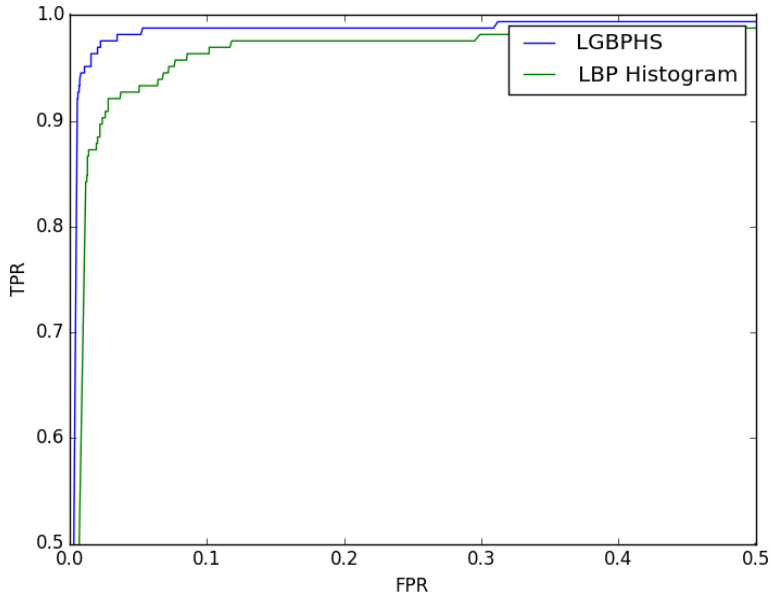


Figure 17: Result of LGBPHS and LBP Histogram

Results. The ROC curves are shown in Figure 17. The average precision of the models are shown as followed:

	LGBPHS	LBP Histogram
Average Precision	95.76%	85.45%

Analysis. This experiment shows that multi-orientation and multi-scale Gabor filters are significantly effective to improve the performance of the model based on LBP. The effectiveness can be explained from the subsequent robustness analysis of LGBPHS. The following diagram shows the robustness of the different histograms to images with variation in lighting.

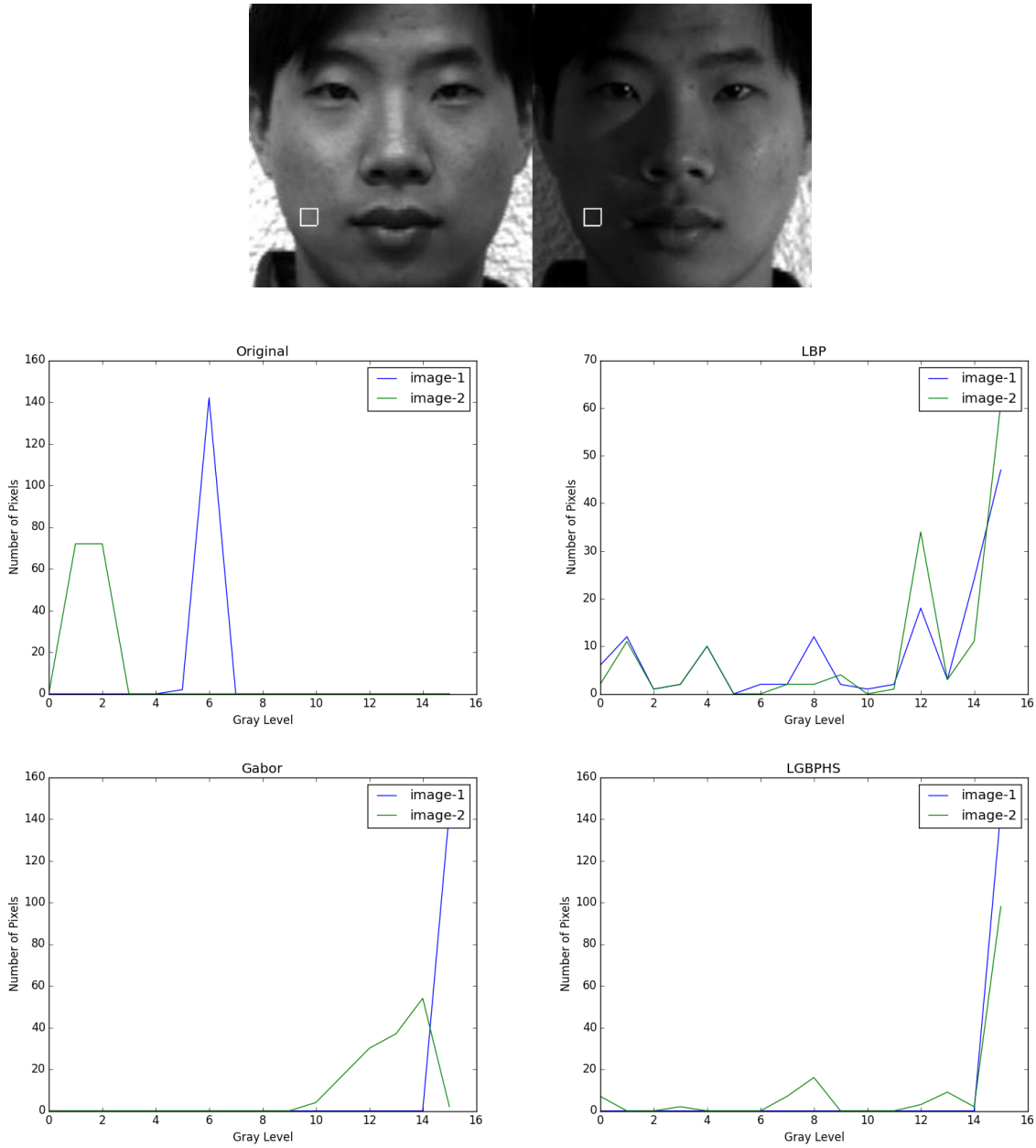


Figure 18: Histogram of LGBPHS

We compare the histograms of 4 representations in Figure 18, extracted from 2 images of the same subject (Yale A subject 04) under different lighting conditions as shown in Figure 18 - Original image intensity, LBP of the image, Gabor magnitude of the image, and the LGBP of the image. A white square as shown in the photos respectively is chose to extract the 4 types of histograms and the results are shown above. By comparing the graph, we the histograms are the most similar of the LGBP of the two photos; thus it explains that LGBP is robust to the lighting variation.

4.9 Experiment 9: Varying LBP algorithms for LGBPHS

Objective. The objective of this experiment is to evaluate the performances of original LBP and circular LBP (i.e. extended LBP) inside LGBPHS algorithm.

Settings. The Gabor Filters of LGBPHS is set to **4** different orientations with **2** different scales for speed considerations. The original LBP is using the default setting since there is no parameter to adjust. The radius of extended LBP is set to **6** with **8** neighbors.

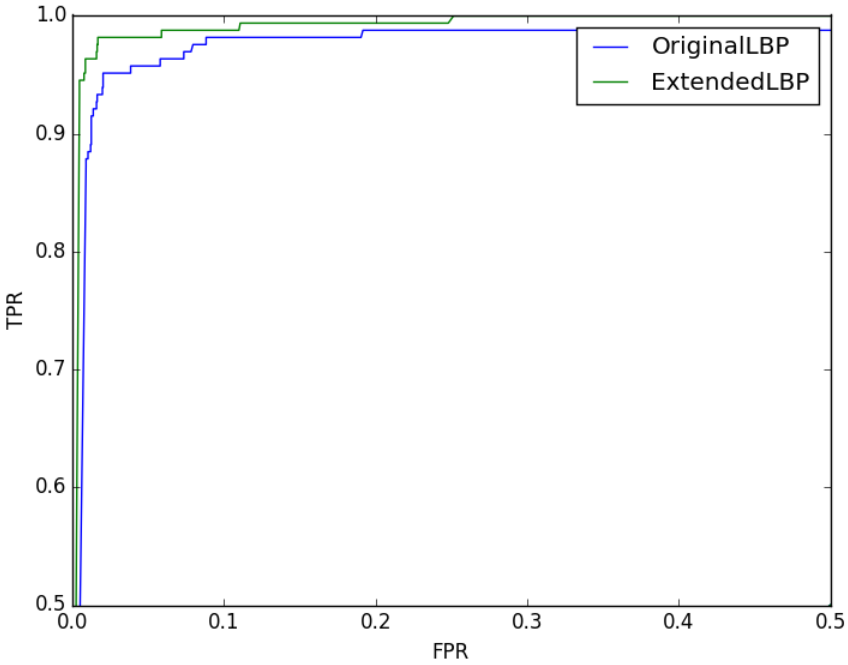


Figure 19: Result of varying LBP

Results. The ROC curves are shown in Figure 19. The average precision of the models are shown as followed:

	LGBPHS with Extended LBP	LGBPHS with Original LBP
Average Precision	95.76%	92.12%

Analysis. This experiment shows that Extended LBP has more freedom to fit the training data by the two additional parameters - radius and number of neighbors. Sometimes it is not effective to choose only the immediate neighbors of a pixel to extract the local features as the original LBP does. The radius should be adjusted according to the dataset.

4.10 Experiment 10: Varying number of scales for LGBPHS

Objective. The objective of this experiment is to evaluate how the number of scales in the Gabor Filter affects the performance of LGBPHS.

Settings. The circular LBP (i.e. extended LBP) of LGBPHS is set to be **3** for radius and **8** neighbors. The Gabor Filters of LGBPHS is set to **4** orientations with varying number of scales as follow:

number of scales: {**1, 5, 9**}

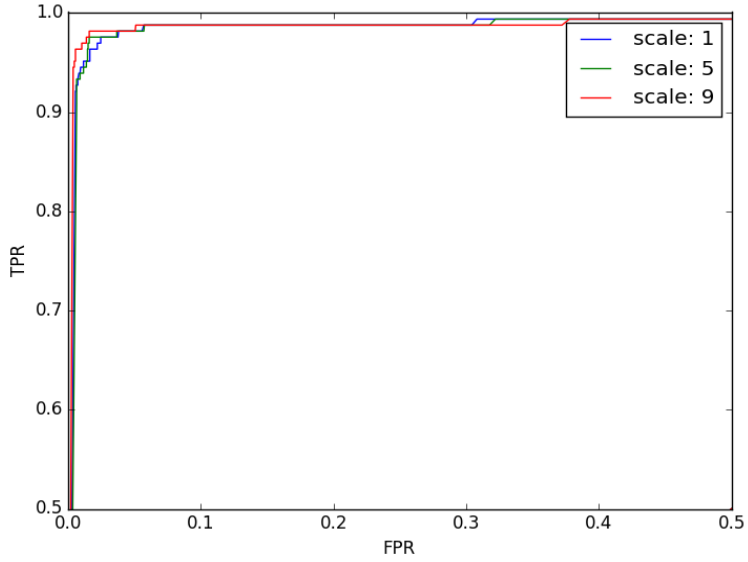


Figure 20: Result of varying number of scales for LGBPHS

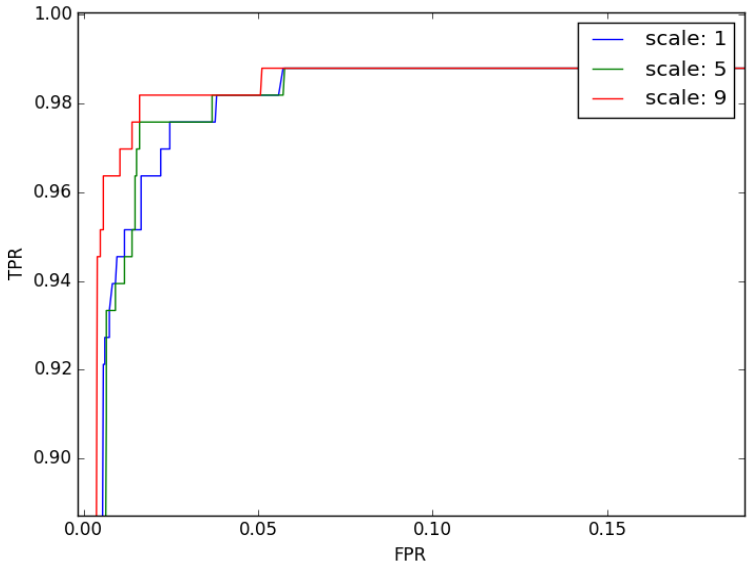


Figure 21: Zoomed in result of varying number of scales of LGBPHS

Results. The ROC curves are shown in Figure 20 and the zoomed in version in Figure 21. The average precision of the models are shown as followed:

Number of Scales	1	5	9
Average Precision	94.55%	95.15%	95.76%

Analysis. By zooming the ROC curves shown in Figure 21, we can find that increasing the number of scales slightly increase the performance of the LGBPHS. It can be explained that increasing the number of scales can increase the number of GMP Maps; thus the model can learn a more sophisticated features of the input face image. However, increasing the number of scales increasing the time complexity of the model and the payoff is marginal.

4.11 Experiment 11: Varying number of orientations for LGBPHS

Objective. The objective of this experiment is to evaluate how the number of orientations in the Gabor Filter affects the performance of LGBPHS.

Settings. The circular LBP (i.e. extended LBP) of LGBPHS is set to be **3** for radius and **8** neighbors. The Gabor Filters of LGBPHS is set to **2** scales with varying number of orientations as follow:

number of orientations: {**2, 5, 8**}

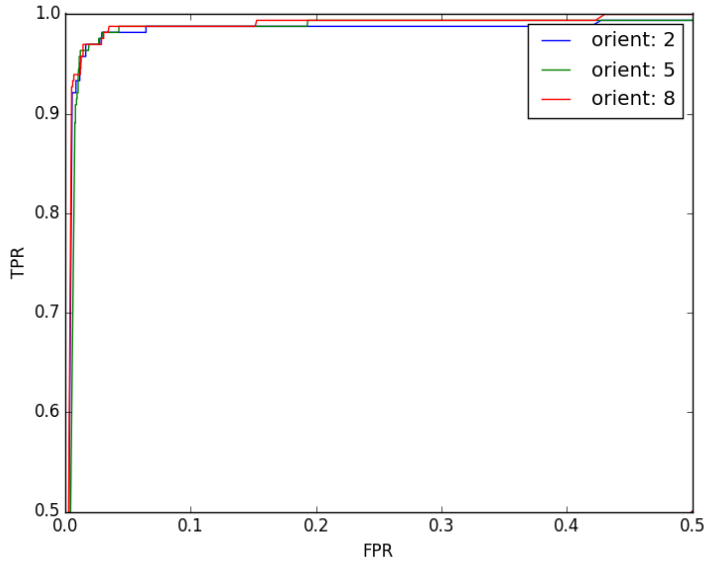


Figure 22: Result of varying number of orientations of LGBPHS

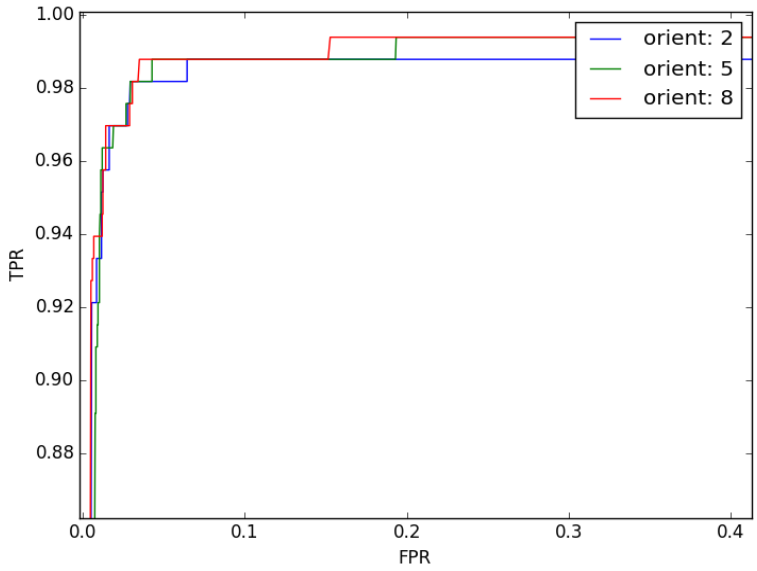


Figure 23: Zoomed in result of varyin number of orientation of LGBPHS

Results. The ROC curves are shown in Figure 22 and the zoomed in version in Figure 23. The average precision of the models are shown as followed:

Number of Orientations	2	5	8
Average Precision	95.15%	95.76%	95.76%

Analysis. By zooming the ROC curves shown in Figure 23, we can find that increasing the number of orientations slightly increase the performance of the LGBPHS. Similar to increasing the number of scales, it can be explained that increasing the number of orientations can increase the number of GMP Maps; thus the model can learn a more sophisticated features of the input face image. However, there is no significant difference in performance by varying number of orientations. Increasing the number of orientations increasing the time complexity of the model and the payoff is marginal.

4.12 Experiment 12: Ensemble LBP Fisherface

Objective. The objective of this experiment is to compare the performances between Ensemble LBP Fisherface and the normal Fisherface algorithms.

Settings. The experiment is carried out by fixing the number of components analyzed in all Fisherfaces to be **14**. The Ensemble LBP Fisherface has an ensemble of circular LBPs of radii: {**3, 6, 10, 11, 14, 15, 19**} in pixel. The single LBP Fisherface has a radius of **11** in pixel.

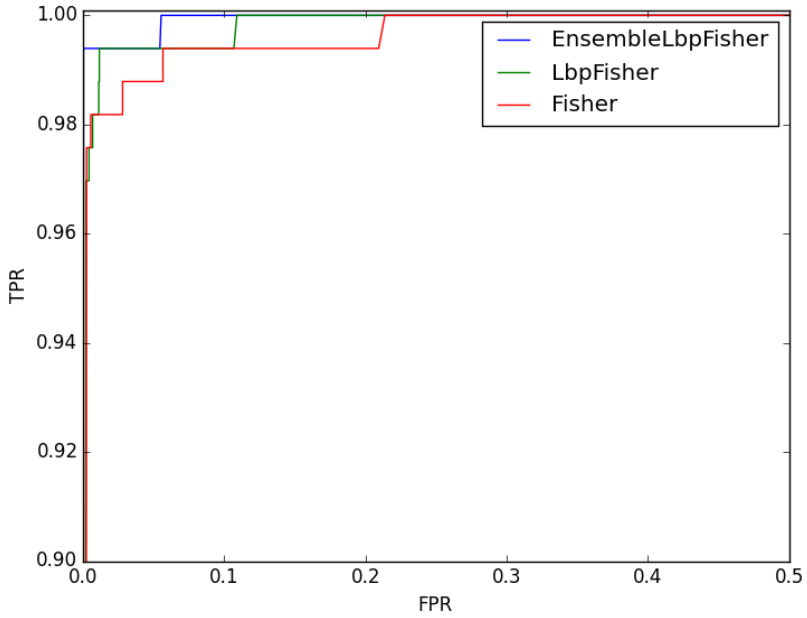


Figure 24: Result of ensemble LBP Fisherface

Results. The ROC curves are shown in Figure 24. The average precision of the models are shown as followed:

	Ensemble LBP Fisher	LBP Fisher	Fisherface
Average Precision	98.79%	97.58%	93.94%

Analysis. First of all, LBP pre-processes the images will increase the performance of Fisherface, which can be explained that LBP operator is robust against monotonic gray scale transformations; therefore LBP Fisher is robust against variations of gray scale conditions for the face images in Yale A dataset.

Second, the results show the ensemble approach effectively increases the performance. By learning the feature vector from multiple representations extracted by LBP algorithm, Fisherface can build multiple models. A more reliable classifier can be obtained by combining the output of multiple component experts. By leveraging the majority voting, the ensemble learner will effectively reduce the error rate.

5 CONCLUSION

In this project, we explored face recognition using kNN classifier with multiple models: PCA, kernel PCA methods, Fisherface (PCA + LDA), NDA, LBP, Gabor Filters, and LGBPHS. Our conclusions can be summarized as follows:

1. The kNN classifier works best with small numbers of k. This is because the data points do not form distinct clusters in the Euclidean space. Hence classifying with more neighbours do not help improve the precision. kNN is most likely to rely on 1 image of the person that is very similar to the testing photo.
2. Increasing the fold number in cross validation increase the precision. This makes sense as increasing the fold number is effectively increasing the number of training data.
3. Among all kernel methods attempted, polynomial and cosines yield the best performance. In using multi-degree polynomial kernel for PCA, we found that degree one (linear) performs better than higher degree. Tuning other parameters does not make large difference on the performance.
4. Fisherface achieved reasonable performance. Its performance is at its highest when number of PCA components is at its upper limit. Replacing the LDA model in Fisherface by NDA improves the performance slightly.
5. The original LBP histogram sequence achieved 85.45% precision. When applying multi-scale multi-orientation Gabor filters, LGBPHS improves LBP histogram sequence by 10.31%.
6. Using LGBPHS with extended LBP instead of original LBP further improves the LGBPHS precision by 3.64%.
7. By tuning the parameters of Gabor, we found that while increasing scales and number of orientation improves the performance slightly, it is a trade-off between performance and time complexity.
8. Using LBP as preprocessing for Fisherface increases the precision of Fisherface by 3.64%.On this basis, ensemble Fisherface with 7 different LBP radii further improved the precision by 1.19%. Ensemble Fisherface achieves a precision of 98.79% on Yale A dataset by the ensemble model.

Although we have achieved reasonable precision on Yale A database, this database is well pre-processed and photos are taken under similar perspective for each subject. In real life, the variation of ambient light and shooting angles and angle may cause further complexity in the face recognition tasks.

6 REFERENCES

[1] The ORL Database of Faces:
http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[2] The Yale Faces Database: http://vision.ucsd.edu/content/yale-face-database

[3] Sebastian Raschka, Step by Step PCA:
http://sebastianraschka.com/Articles/2014_pca_step_by_step.html#transform

[4] Bernhard SchSlkopf, Alexander Smola, Klaus-Robert Mfiller, "Kernel Principal Component Analysis" 1998

[5] Scikit learn metrics: http://scikit-learn.org/stable/modules/metrics.html

[6] Fisherfaces: http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf

[7] Wenchao Zhang, Shiguang Shan, Wen Gao, Xilin Chen, Hongming Zhang "Local Gabor Binary Pattern Histogram Sequence (LGBPHS): A Novel Non-Statistical Model for Face Representation and Recognition": http://www.jdl.ac.cn/user/sgshan/pub/ICCV2005-ZhangShan-LGBP.pdf

[8] Philipp Wagner, "Fisherfaces": http://www.bytefish.de/blog/fisherfaces/#introduction