

CS 479/679: Pattern Recognition

Programming Assignment 3

Gunner Stone

Derek Stratton

We did approximately 50% of the coding and 50% of the writeup each.

## 1. Theory

The underlying theory behind Principal Component Analysis (PCA) is to reduce the dimensionality of a dataset while keeping as much variation in the data as possible. This essentially boils down to removing dimensions in the data with little to no variation, while keeping the dimensions with high variation. Once these dimensions of high variation are found (also known as the principal components), the original data is mapped to lie only along the principal components.

In this project's implementation, PCA is a dimensionality reduction technique that allows a given face image vector to be represented by a small list of weight coefficients. The images used in this project were from the FERET dataset. The high definition faces were matrices of size 48x60 while the low definition faces were 16x20.

These coefficient values mean nothing without their corresponding principal components, better known as eigen faces. Eigen faces are found through a lengthy and computationally heavy process involving computing the eigenvectors of the covariance matrix of the entire image training set. This covariance matrix, when found through conventional means, is very large 2880x2880. This means calculating the eigenvectors cannot be done within a reasonable amount of time or space.

A covariance matrix of the training data is calculated by computing  $\mathbf{A} \cdot \mathbf{A}^T$ . Matrix  $\mathbf{A}$  is created by compiling all flattened training faces as the columns of a new matrix  $\mathbf{A}$ . It is important that these flattened face vectors in  $\mathbf{A}$  are centered. Centering a face is accomplished by subtracting the sample mean face vector from each face.

A trick employed in our project, and also in the original eigenfaces paper, is instead of finding the eigenvectors of the covariance matrix  $\mathbf{A} \cdot \mathbf{A}^T$ , instead calculate  $\mathbf{A}^T \cdot \mathbf{A}$ , find the eigenvectors of this result (called  $\mathbf{v}$ ), and map it back to the original input space by multiplying it with  $\mathbf{A}$  as seen here:  $\mathbf{A} \cdot \mathbf{v}$ . This shortcut allows your 'covariance matrix' to be significantly reduced in size. This ultimately allows for the eigenvectors in lower dimensional space  $\mathbf{v}$  to be calculated more efficiently.

This shortcut doesn't come without its downsides though. By employing this shortcut, only the top  $M$  eigenfaces can be found where  $M$  is the number of training samples. But since our goal is dimensionality reduction, this isn't a problem, because we want a good reconstruction using even less than  $M$  eigenfaces.

The one great thing about this long and arduous process is that it only has to be done a single time for a given training set. Once the eigen faces are found, the coefficients of an input image can be found by applying a dot operation from an input image onto its corresponding eigenfaces. This will result in a list of coefficients/weights that tell the programmer how much each eigenface should be weighted when trying to reconstruct the input image.

The important thing to remember when doing this is to center your training data by calculating the mean vector and subtracting it from all inputs.

PCA will result in higher quality reconstructions if mapped using a greater number of principal components. Because of this, a designer can selectively choose the number of principal components used during mapping, making a tradeoff between compression and quality.

This project primarily seeks to examine the tradeoff between the number of principal components used and facial recognition accuracy.

The facial recognition portion of this project works by comparing the weights of an input image and weights of training images. Ideally, similar faces would have very similar weights and non-similar faces would have weights very different from each other. A facial 'similarity score' is calculated by taking the mahalanobis distance between an input image weights and all of the training image weights. The best match is the smallest mahalanobis distance.

However, the best match might not always be correct. If an image of a trout was plugged in as the input image, PCA would do its best to represent the trout as a linear combination of eigenfaces. The trout's eigenface weights will be compared with those in our training set and a 'best match' will be found. It won't be a very good match, but it'll be the best one there is. This is exactly the reasoning behind setting a threshold for that mahalanobis distance to be considered a match at all. If the 'best match' falls below a threshold, it is not only the best match but also a good match. Without a threshold, trouts could impersonate high level security faculty and threaten national security.

## 2. Implementation

Our project was coded in C++ using the Eigen and OpenCV libraries. We read in all the training images as `MatrixXds` and stored them all in a C++ vector. From here, the mean training face was calculated (shown in Results and Discussion section). This mean was subtracted from each of the training images to give centered images. Each of these centered images were flattened into a Vector and compiled into the columns of matrix **A**. The shortcut covariance matrix was found by  $\mathbf{A}^T \cdot \mathbf{A}$ . Our code called this result **s**. The eigenvalues and eigenvectors were found for **s** using the Eigen library's `SelfAdjointEigenSolver` function. Despite being a smaller covariance matrix, this still took a decent amount of time on our computers. These eigenvectors are not in the correct dimensions though (only being  $1 \times M$ ). To get them to be sized  $1 \times N^2$ , we had to multiply  $\mathbf{A} \cdot \mathbf{s}$ . This resulted in our top M eigenfaces flattened as the columns of this resultant matrix.

These eigenfaces and their corresponding eigenvalues were saved as raw floats to a file using `opencv's FileStorage` object. Saving these as files allowed us to avoid retraining PCA every time we wanted to run our code.

To get a given input image in terms of its weights, simply perform the dot product for an input image and a given eigenface. Do this for K eigenfaces to get K weights. From here we were

able to calculate the best matches between our testing set B and our training set A. Matches between testing and training images were performed using the Mahalanobis distance on both of the respective weight vectors. Smaller Mahalanobis distance indicates a closer match. The best match was the test/training pairs that had the lowest 'score'.

If the actual correct match was within the top N best matches (N was a hyperparameter), then we recorded it in our graph shown in the results and discussion section. We performed this data analysis for the top K eigenfaces that preserved 80%, 90%, and 95% of the data respectively.

Additionally, we also implemented a threshold value that our best match score could be compared against. If the best match score was below the threshold, it was counted as being a valid match. Likewise if the best match score was above the threshold, it would be counted as an intruder. This threshold value is an optimizer a designer can select to allow for more leniency or strictness in this recognizer. For this assignment, we were asked to use a ROC graph to find the threshold that optimized for the same number of false positives and true positives.

We additionally performed the implementation above but for low resolution images as well. Their results and discussion about the differences between them are noted below.

Throughout writing the software, we also performed various checks at various steps as discussed in the assignment. We checked for eigenvectors being unit length, covariance metric symmetry, reconstruction of training images with low error, and recognizing a training image by finding a zero distance in face space.

### 3. Results and Discussion

#### a. Experiments on fa\_H training and fb\_H testing sets

##### i. Image Samples



Fig. 1: Image of the Average Face in the fa\_H training set.



Fig. 2: Images of the Eigenfaces corresponding to the largest 10 eigenvalues.

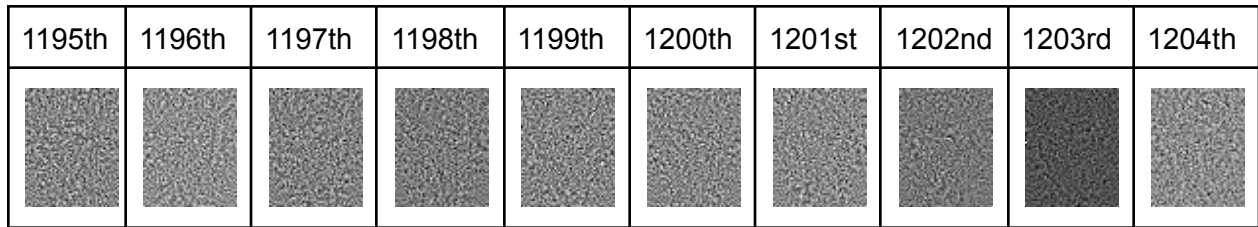


Fig. 3: Images of the Eigenfaces corresponding to the smallest 10 eigenvalues.

ii. Face Recognition Quantitative Results for 80%, 90%, and 95% Preservations (Includes repeats for Part a.V)

Figure 4 shows the high definition comparative CMC graph for the performance (accuracy) of finding a correctly matching face within the top N returned matches for different amounts of data preservation. Data preservation, as asked for in the homework, describes the amount of data preserved using PCA's data reduction. This graph shows 80%, 90%, and 95% preservation.

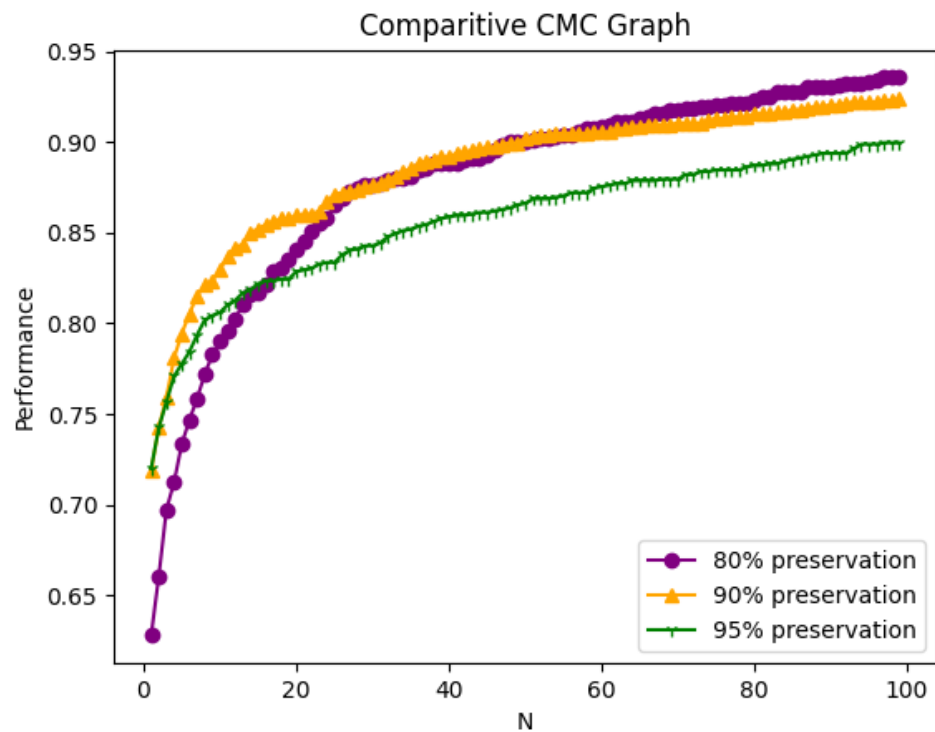


Fig. 4: Comparison of CMC Curves for 80%, 90%, and 95% Data Preservations on hd images for Varying N

iii. Face Recognition Correct Matches for 80%, 90%, and 95% Preservations  
(Includes repeats for Part a.V)



















80% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
90% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
95% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					

Fig. 5: Three Correct Matches for Varying Levels of Information Preservation.

From 80% preservation you expect to see a 'better' match at 95% preservation, but actually it is just the same best match from the 80% preservation. This indicates that having more information preservation will not influence the recognizer's top match.

iv. Face Recognition Incorrect Matches for 80%, 90%, and 95% Preservations (Includes repeats for Part a.V)



















80% information preservation; N=1					
Input Image 1	Incorrectly Matched Img 1	Input Image 2	Incorrectly Matched Img 2	Input Image 3	Incorrectly Matched Img 3
					
90% information preservation; N=1					
Input Image 1	Incorrectly Matched Img 1	Input Image 2	Incorrectly Matched Img 2	Input Image 3	Incorrectly Matched Img 3
					
95% information preservation; N=1					
Input Image 1	Incorrectly Matched Img 1	Input Image 2	Incorrectly Matched Img 2	Input Image 3	Incorrectly Matched Img 3
					

Fig. 6: Three Correct Matches for Varying Levels of Information Preservation.

From looking at the above table, you can see that although the people are different, the lighting in both the input and incorrectly matched images are very similar. This agrees with many papers that highlight the shortcomings of PCA in its inability to work with different lighting conditions. You can see this issue in the top 10 eigenfaces being really sensitive to lighting & light direction.

v. Analysis of Results for Varying Information Preservations

Redoing the experiments for 90 and 95% preservation are incorporated in all of the above deliverables from 3a<sub>ii</sub> - 3a<sub>iv</sub>. Interestingly, while the higher preservation values outperform for lower N, the 80% preservation achieved better results for higher values of N. A possible reason for this is overfitting to the training set with too many components.

b. Experiments on subsets of fa\_H training and fb\_H testing sets for intruder detection

In this experiment, a subset of the fa\_H set was used for training, and the distances in face space with the test set were compared with a threshold to determine if faces were intruders or not. The ROC curve of threshold values between our minimum and maximum distances are shown in Fig. 7, along with a baseline plot of a line with slope 1. The corresponding thresholds that yielded values near the threshold were at about  $T = 0.1$ .

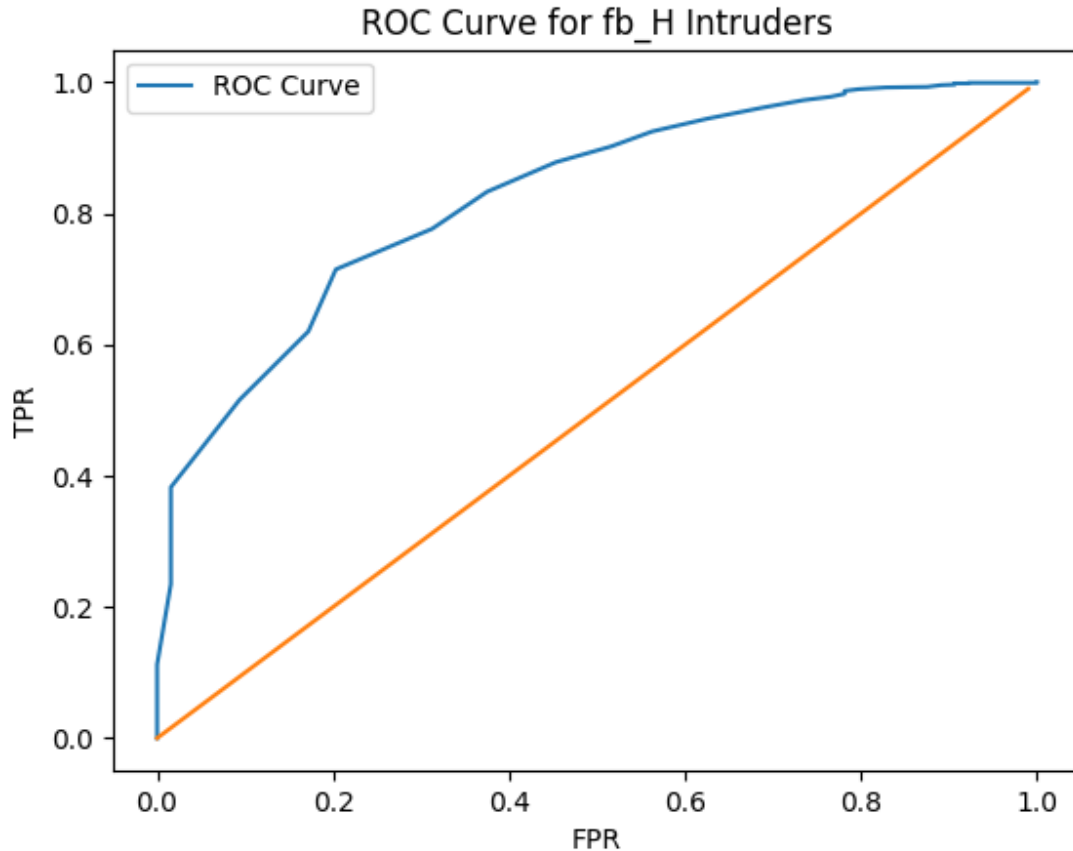


Fig. 7: ROC Curve for Intruder Detection in fb\_H.

c. Experiments on fa\_L training and fb\_L testing sets

i. Image Samples



Fig. 8: Image of the Average Face in the fa\_H training set.



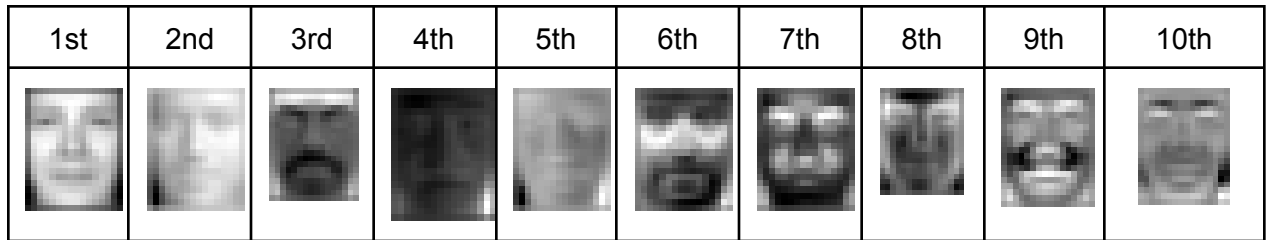


Fig. 9: Images of the Eigenfaces corresponding to the largest 10 eigenvalues.

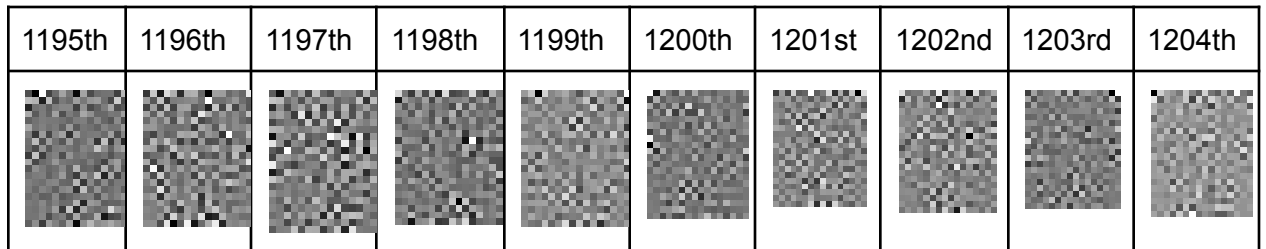


Fig. 10: Images of the Eigenfaces corresponding to the smallest 10 eigenvalues.

## ii. Graph

Figure 11 shows the low definition comparative CMC graph for the performance (accuracy) of finding a correctly matching face within the top N returned matches for different amounts of data preservation. Data preservation, as asked for in the homework, describes the amount of data preserved using PCA's data reduction. This graph shows 80%, 90%, and 95% preservation.

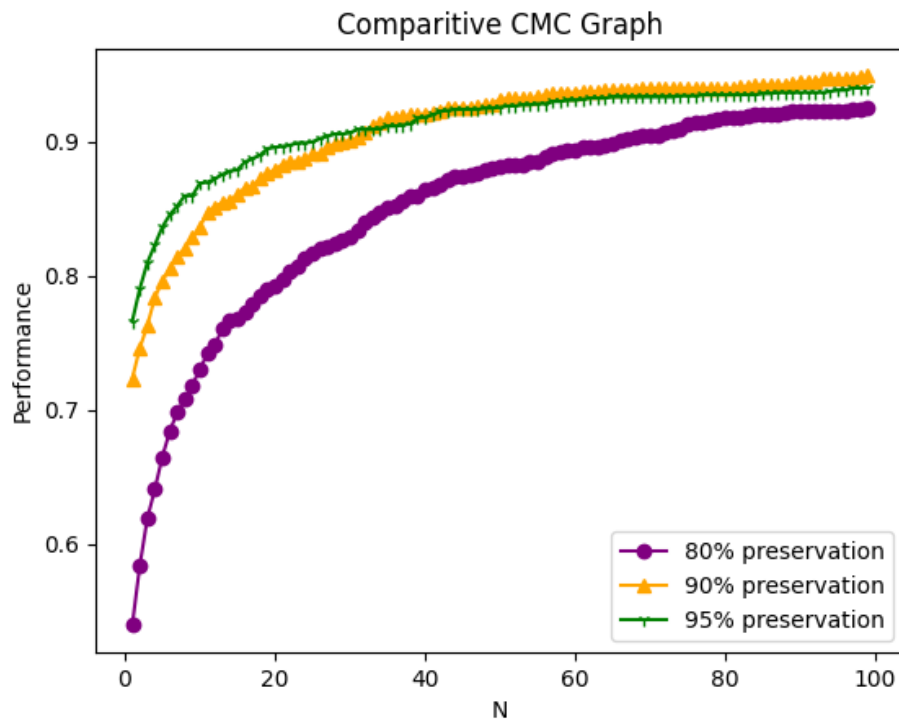


Fig. 11: Comparison of CMC Curves for 80%, 90%, and 95% Data Preservations on Id images for Varying N

iii. Recognition correct matches 80%, 90%, 95%














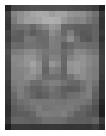

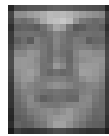


80% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
90% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
95% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					

Fig. 12: Three Correct Matches for Varying Levels of Information Preservation.

iv. Recognition incorrect matches 80%, 90%, 95%
















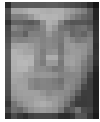


80% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
90% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					
95% information preservation; N=1					
Input Image 1	Correctly Matched Img 1	Input Image 2	Correctly Matched Img 2	Input Image 3	Correctly Matched Img 3
					

Fig. 13: Three incorrect Matches for Varying Levels of Information Preservation.

d. Experiments on subsets of fa\_L training and fb\_L testing sets for intruder detection

In this experiment, a subset of the fa\_L set was used for training, and the distances in face space with the test set were compared with a threshold to determine if faces were intruders or not. The ROC curve of threshold values between our minimum and maximum distances are shown in Fig. 14, along with a baseline plot of a line with slope 1. The corresponding thresholds that yielded values near the threshold were at about  $T = 0.03$ .

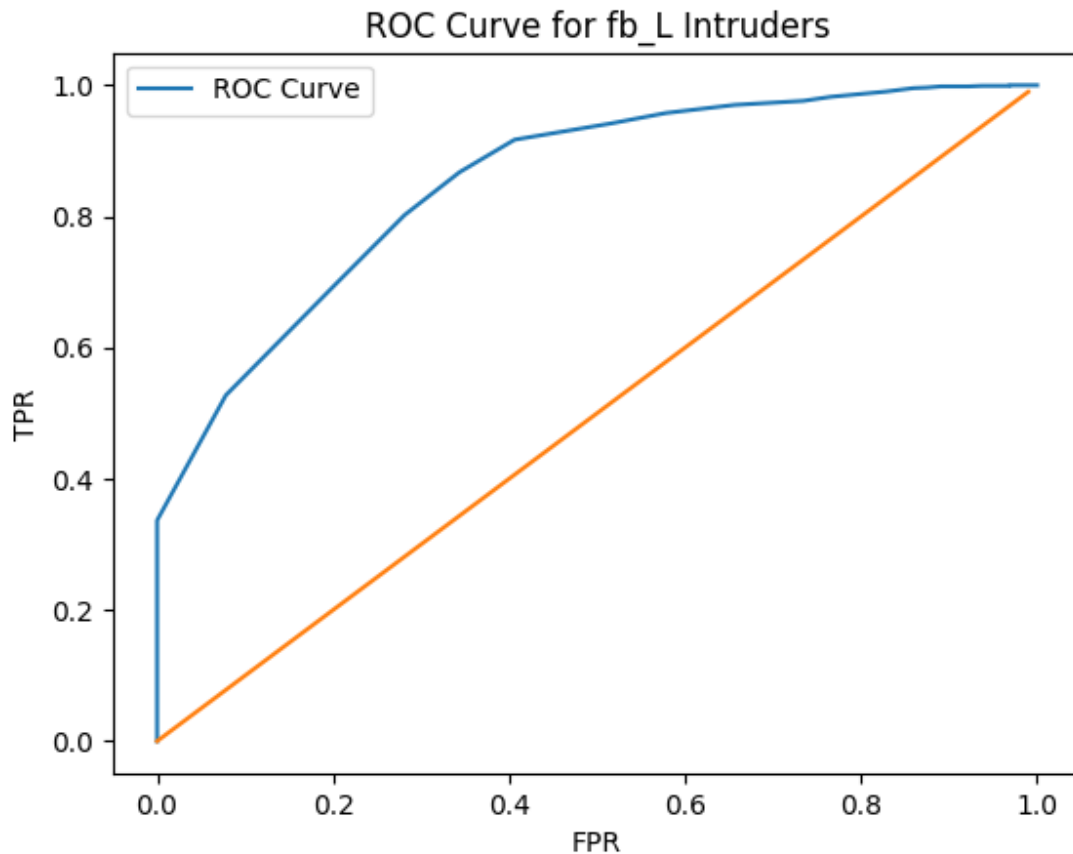


Fig. 14: Intruder Detection for fb\_L.

#### e. Comparison of Low-Resolution and High-Resolution Experiments

The differences between using low resolution and high resolution to run facial recognition were apparent from the CMC graphs. The high resolution facial recognition started out with much higher accuracy with  $N=1$ . The higher resolution recognizer also converged to a higher accuracy with  $N = 100$ . However, the low resolution converged much faster, hitting its peak accuracies around  $N = 40$  for the 90% and 95% preservations. If top  $N$  accuracy is the goal, high resolution is the better option. If a company wants a budget option or has to work with low resolution images for their cameras, the low resolution recognizer also works, just only look within the top 40 best matches.

For rejecting intruders, both datasets had similar shapes to their ROC curves. However, a good threshold value was about 0.1 for the fb\_H set and about 0.03 for the fb\_L set. The distance threshold for the low resolution set was smaller for rejecting intruders optimally than for the higher resolution set, since the low resolution images caused the distance values to decrease.