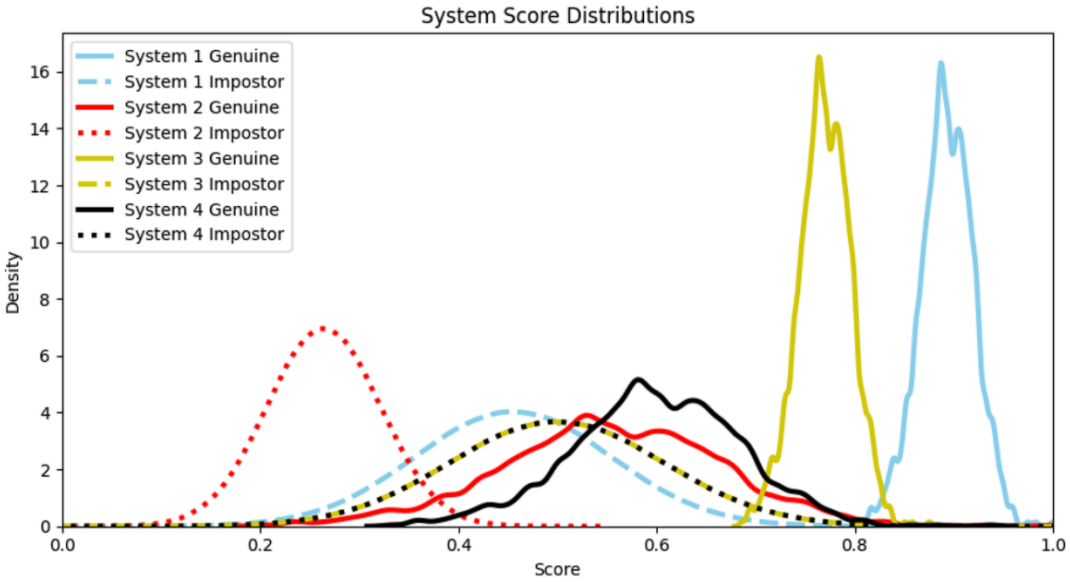
Biometrics Assignment 1 Report
Dimitrios Chatzakis r0977164

Q1: Score Distributions

- **Plot the score distributions**



The plots for each distribution were made by taking all the genuine (genuineID == 1) and impostor (genuineID == 0) cases and plotting them one against the other using the sns.kdeplot. The kdeplot uses a Kernel Density Estimate (KDE) based on Gaussian fitting to create a continuous Probability Density Functions PDFs. It has a bandwidth hyperparameter to specify the width of the Gaussian and calculate the PDF based on:

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

Where n is the number of samples, h the bandwidth hyperparameter and K the Gaussian Kernel. This way a nice smoothed plot can be achieved maintaining the details of the underlying distributions.

- **Do you need to normalize the distributions? Why (not)?**

Yes, you need to normalize the distributions for the visualizations, otherwise the huge discrepancy between the impostor and genuine score sizes would show the impostor histogram peaks, and the genuine peaks would be too low to be seen.

- **Describe qualitatively this combined plot. Which system performs the best based on this plot?**

Based on the plot we can discern that the best system is the first, since there is minimal overlap between the two distributions. Then, more easily discernible plots are between the system 3, 2 and 4 systems in descending order.

- **Which system corresponds to which of the following descriptions?**
    1. **small inter-user distance, small intra-user variation**
       This case is of system 2 since the 2 curves overlap a bit and the genuine distribution is very narrow.
    2. **small inter-user similarity, small intra-user variation**
       This case is of system 1 since curves almost do not overlap much and the genuine distribution is very narrow.
    3. **large inter-user similarity, large intra-user variation**
       This case is of system 4 since both curves overlap quite much and are quite wide
    4. **large inter-user distance, large intra-user variation**
       This case is of system 2 since curves do not overlap quite much and are quite wide.

Q2: ROC Curves

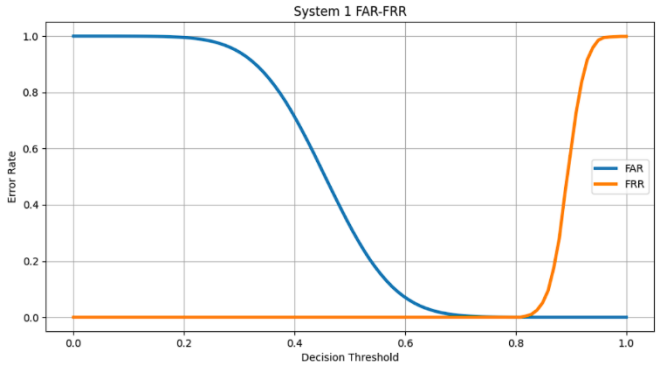
- **Calculate FPR, TPR from the matching scores.**

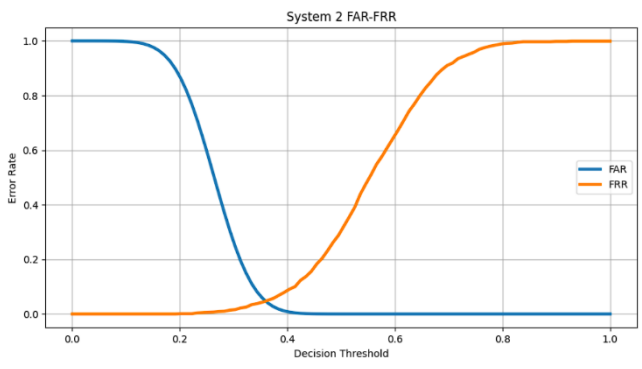
```
"""Calculate FPR, TPR from the matching scores."""
thresholds = np.linspace(0, 1, 100)
# Initialize Lists to store each system list
f1SystemsList, accSystemsList, precSystemsList, recallSystemsList, TPRSystemsList, FPRSystemsList, FRRSystemsList = [], [], [], [], [], [], []
for _, _, genuineID, scores in systems:
    f1Scores, accScores, precScores, recallScores, TPRScores, FPRScores, FRRScores = [], [], [], [], [], [], []
    for threshold in thresholds:
        f1, accuracy, precision, recall, TPR, FPR, FRR = calculateRates(scores, genuineID, threshold)
        # Append ton lists
        f1Scores.append(f1)
        accScores.append(accuracy)
        precScores.append(precision)
        recallScores.append(recall)
        TPRScores.append(TPR)
        FPRScores.append(FPR)
        FRRScores.append(FRR)
    # Append to system List
    f1SystemsList.append(f1Scores)
    accSystemsList.append(accScores)
    precSystemsList.append(precScores)
    recallSystemsList.append(recallScores)
    TPRSystemsList.append(TPRScores)
    FPRSystemsList.append(FPRScores)
    FRRSystemsList.append(FRRScores)
```

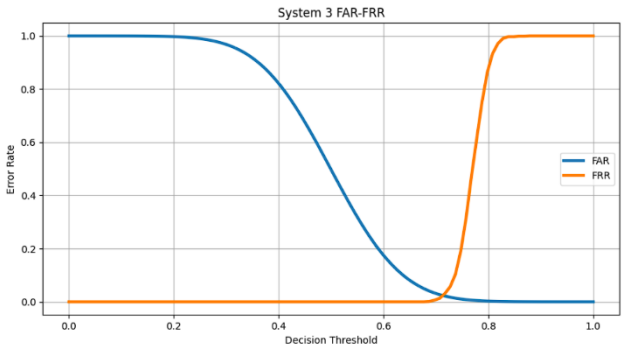
```
def calculateRates(scores, genuineID, thresholds):
    # Calc stats
    TP = sum((scores>=threshold) & (genuineID==1))  # True Positives
    FN = sum((scores<threshold) & (genuineID==1))   # False Negatives
    FP = sum((scores>=threshold) & (genuineID==0))  # False Positives
    TN = sum((scores<threshold) & (genuineID==0))   # True Negatives
    # Calc Rates
    tpr = TP/(TP+FN) if TP+FN>0 else 0
    fpr = FP/(FP+TN) if FP+TN>0 else 0
    frr = FN/(TP+FN) if TP+FN>0 else 0
    # Calc Precision, Recall, Accuracy and F1
    precision = TP/(TP+FP) if TP+FP>0 else 0
    recall = TP/(TP+FN) if TP+FN>0 else 0
    accuracy = (TP+TN)/(TP+FP+FN+TN) if (TP+FP+FN+TN) > 0 else 0
    f1Score = 2*(precision*recall)/(precision+recall) if (precision+recall)>0 else 0
    return f1Score, accuracy, precision, recall, tpr, fpr, frr
```

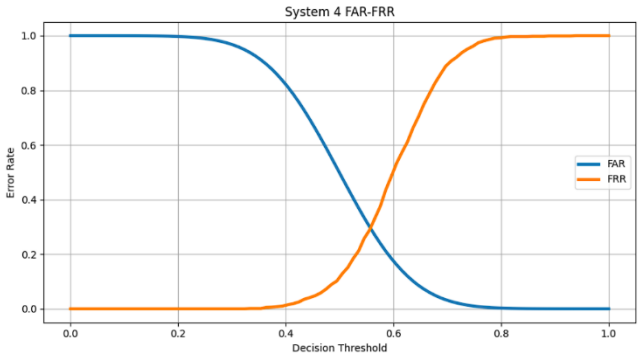
I used the code to the left and above to calculate the True Positive Rate (TPR), False Rejection Rate (FRR), and False Positive Rate (FPR) along with the metrics. I used the threshold as a first filter to get all the scores, and the genuineID to see the classification based on the threshold.

- **Plot FAR and FRR as a function of matching scores**.



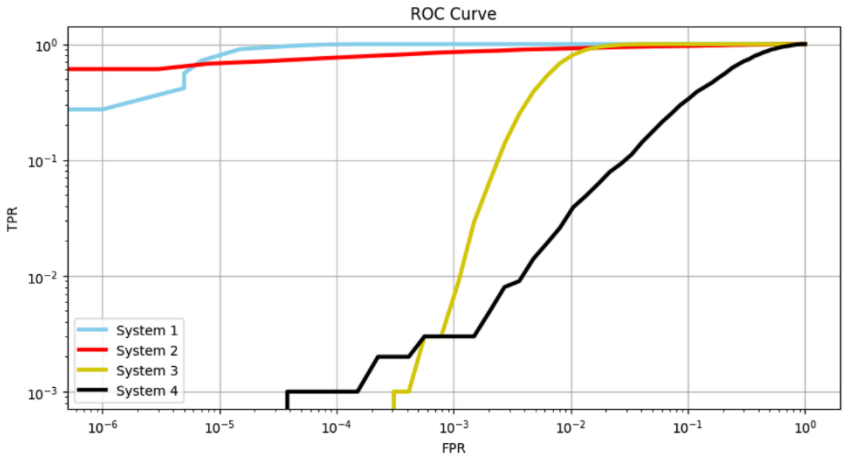






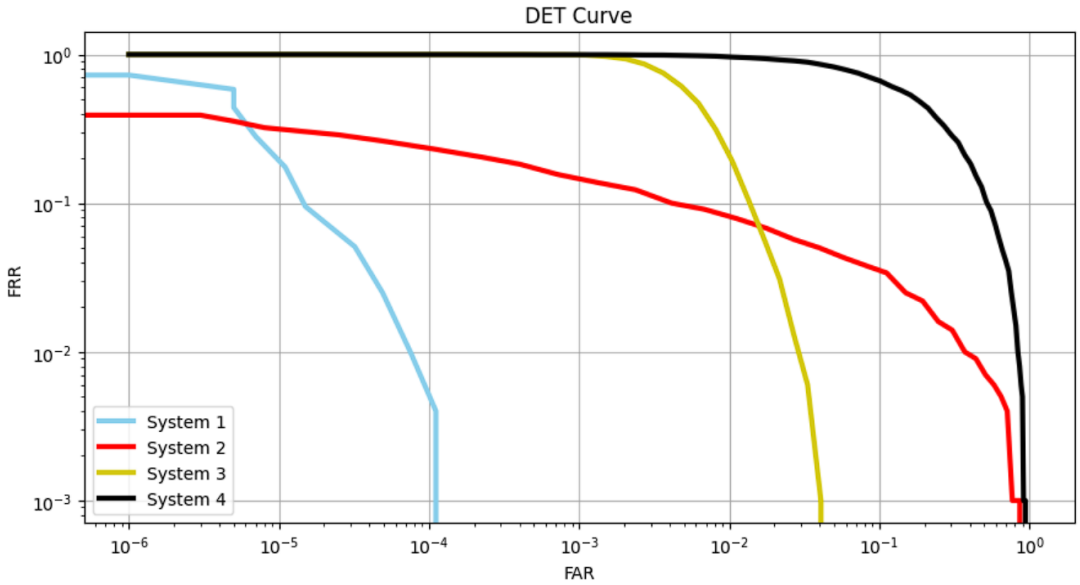
Here we can see again the superiority of the first system and the least performance by the 4th. Again however, it is not clear whether system 2 or 3 us superior with certainty.

- **Plot the ROC curve for the four systems in a single plot. Plot for linear and logarithmic scale if needed. What do you observe?**



I observe that for most values system 1 is superior to the rest as expected. However, there are certain areas where system 2 is better. That is also observed in the DET curve figure with a high FRR initially in the y axis. The system 4 is again performing the worst and system 3 is strictly worse than 1 but in a small band better than 2.

- **Plot the Detection Error Trade-off (DET) curve. How does it compare to ROC?**

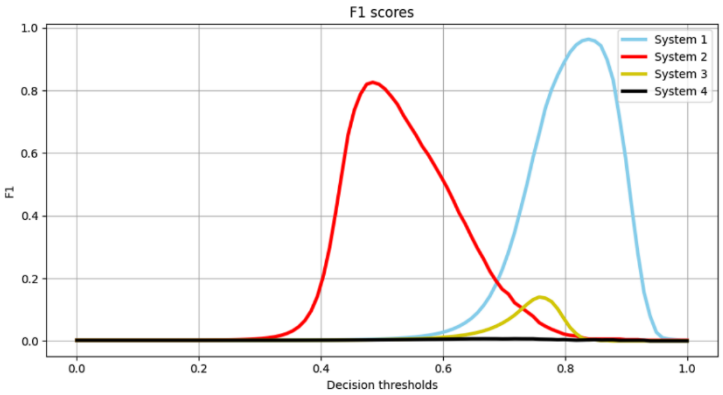


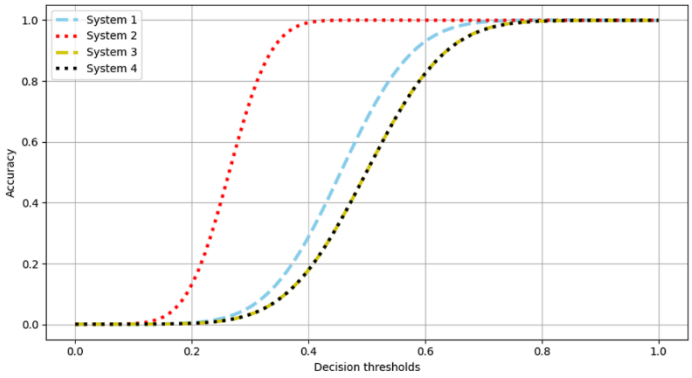
The DET curve shows similar results to the ones in the ROC.

Q3: Classification metrics

- **Plot F1 and accuracy as a function of the decision thresholds on the similarity score for the four systems in a single plot.**

We can see the optimal results for f1 are for the 1st system, continuing to system 2, 3 and finally 4. This metric is much more robust due being robust to class imbalance since the true positives are in the numerator of precision and recall. In contrast in accuracy system 2 appears better than the rest with system 3,4 having same results





- **Calculate the threshold and accuracy for which F1 is maximal. Is it an interesting operating point?**

```
For system 1 the max f1 is 0.963 for the threshold 0.84 and accuracy 1.000
For system 2 the max f1 is 0.826 for the threshold 0.48 and accuracy 1.000
For system 3 the max f1 is 0.140 for the threshold 0.76 and accuracy 0.992
For system 4 the max f1 is 0.007 for the threshold 0.72 and accuracy 0.978
```

It is indeed an interesting point for operating where we can see that the accuracy is high in all cases while the better f1 is not.

- **Do the same for the classification error (accuracy). Is there a difference?**

```
For system 1 the max accuracy is 1.000 for the threshold 0.84 and f1 0.963
For system 2 the max accuracy is 1.000 for the threshold 0.48 and f1 0.826
For system 3 the max accuracy is 0.999 for the threshold 1.00 and f1 0.000
For system 4 the max accuracy is 0.999 for the threshold 1.00 and f1 0.000
```

Here we can see that despite high accuracy the f1 score can be very low. Thus, accuracy in this case is not a good metric.

- **Is accuracy a good performance metric in this case?**

In an imbalanced dataset, the majority class dominates the metric, meaning a model can achieve high accuracy simply by predicting every instance as the majority class, while completely misclassifying the minority class. It also does not distinguish between types of errors. It treats false positives and false negatives equally, although in many scenarios, the cost of these errors can be vastly different.

Q4: AUC, EER and alternatives

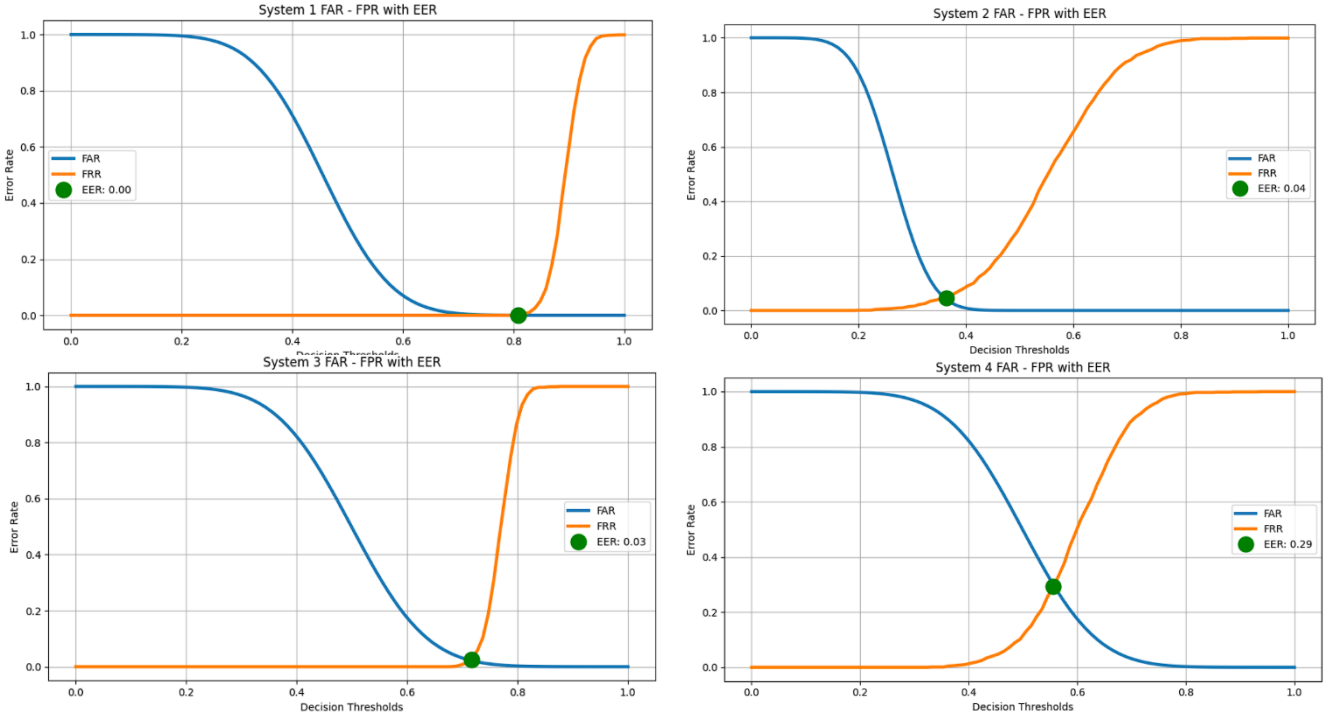
- **Calculate ROC AUC. Is this a good metric? What does it reveal about the system?**

```
from sklearn.metrics import roc_auc_score
for i, (_, _, genuineID, scores) in enumerate(systems):
    print(f"The ROC AUC for system {i+1} is {roc_auc_score(genuineID, scores):.4f}")
```

```
The ROC AUC for system 1 is 1.0000
The ROC AUC for system 2 is 0.9868
The ROC AUC for system 3 is 0.9927
The ROC AUC for system 4 is 0.7691
```

The ROC AUC is a good metric for finding the best system overall, however, it is a scalar and as such does not show the overall behavior of the system in detail. Also, it does not allow for imbalanced data as robustly as PR curves. This shows in the fact that system 2 appears slightly worse than system 3.

- **Calculate (by approximation) the EER and plot it on the FAR-FRR curve. Is this a good peration point?**




```
# Get differences
diffs = np.abs(np.array(FPRList[i])-np.array(FRRList[i]))
# Find min index
minIdx = np.argmin(diffs)
# Average at closest point
EER = (FPRList[i][minIdx]+FRRList[i][minIdx])/2
plt.plot(thresholds, FPRList[i], label='FAR')
plt.plot(thresholds, FRRList[i], label='FRR')
# Mark EER
plt.plot(thresholds[minIdx], EER, 'go', label=f'EER: {EER:.2f}')
```

The EER was calculated based on the code above. It shows the optimal threshold for operation for the system if security and convenience are equally important, otherwise not.

- **Calculate the decision threshold for which the sum of FRR and FAR is minimal. Is this point similar to the total classification error?**

This point could be like the total classification error but it does not consider True Negatives, therefore should be used with caution. We can compare and see that the two points differ a lot in the cases of systems 3, 4.

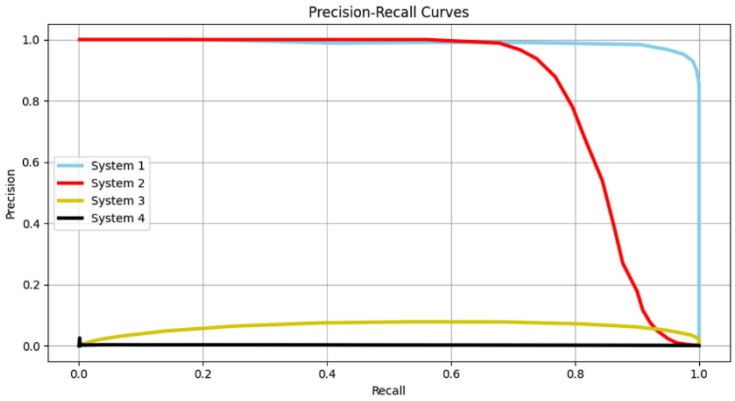
```
The minimal sum of FRR and FAR is 0.000 for the threshold 0.81
The minimal sum of FRR and FAR is 0.084 for the threshold 0.37
The minimal sum of FRR and FAR is 0.039 for the threshold 0.70
The minimal sum of FRR and FAR is 0.581 for the threshold 0.54
```

- **Discuss the importance of the FRR and the FAR if you want a very secure versus a very convenient system.**

In high-security environments, a lower FAR is preferred, even if it means a higher FRR. Conversely, in more user-friendly settings, a lower FRR might be prioritized to reduce the inconvenience to users, even at the cost of a slightly higher FAR. The first case would be identification for your bank of phone while in the second would be for admission to a concert.

Q5: Precision-Recall curves and related summary measures

- **Calculate and plot the Precision-Recall curve for these systems. What does it reveal about the performance of the systems?**



The difference is that the PR curve is way more robust for imbalanced datasets such as ours due to the inherent robustness of precision and recall by focusing in the positive class in the numerator. This reveals the clear superiority of system 1, 2, 3, 4 in descending order, given that precision and recall are equally important.

- **Does the ROC or the PRC make more sense to evaluate these four systems?**

For the reason stated above the PRC would be more important. In applications where the cost of false negatives is high, focusing on recall becomes important, while in applications where the cost of false positives is high, precision is crucial. The PRC directly addresses these aspects, making it highly relevant for critical applications dealing with imbalanced data.

- **Calculate the Area Under the PR-curve. Discuss.**
- **Calculate the average precision scores. Discuss its value.**

```
System 1 Area Under the PR Curve: 0.9883
System 2 Area Under the PR Curve: 0.8392
System 3 Area Under the PR Curve: 0.0632
System 4 Area Under the PR Curve: 0.0028
```

```
System 1 Average Precision Score: 0.9893
System 2 Average Precision Score: 0.8405
System 3 Average Precision Score: 0.0637
System 4 Average Precision Score: 0.0028
```

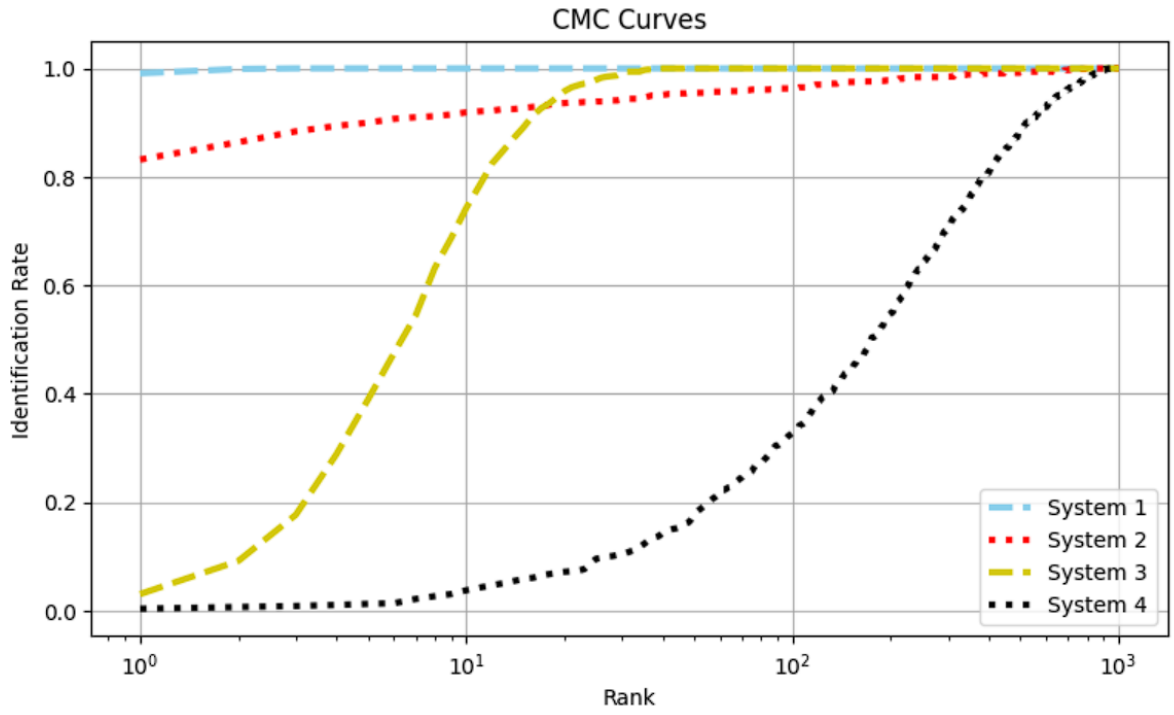
We see here that the AUC under the PR pronounces the differences between the systems in a lot more robust way, noting the huge differences in values. Average Precision scores are very similar noting the robustness of the model to the imbalanced dataset. A discrepancy between AP and PRC AUC might indicate the model performs inconsistently across different recall levels. For example, a model might have very high precision at low recall levels (capturing a small number of positives very accurately) but then significantly lower precision as recall increases. Such a pattern might lead to a higher PR AUC if the precision drops are gradual, but a lower AP if the precision drops sharply at thresholds that greatly increase recall.

Q6: CMC curves

Calculate the Cumulative Matching Characteristic curve (implement this yourself)

```python
def calcCMC(similarityMatrix):
    # Number of individuals for comparisons
    numQueries = similarityMatrix.shape[0]
    # Initialize rank-storing array
    cmcCounts = np.zeros(numQueries)
    for i in range(numQueries):
        # Sort scores for the i-th query in descending order
        sortedIdx = np.argsort(similarityMatrix[i])[::-1] # [::-1] reverses it
        # Find the rank of the genuine score in the diag elmnt
        rank = np.where(sortedIdx == i)[0][0]
        # Increment the CMC count for all positions from the genuine rank onwards
        cmcCounts[rank:] += 1
    # Normalize by the number of queries (elmnts in row) to get probabilities
    cmcCurve = cmcCounts / numQueries
    return cmcCurve
```

The CMC curve plots the probability of correct identification on the y axis against the rank in the x. The rank refers to the position of the correct match within a sorted list of all candidates, ranked according to their similarity to the input sample. For instance, a rank-1 identification rate would indicate the percentage of times the correct individual was the top match. Similarly, a rank-5 identification rate would show the percentage of times the correct match was within the top 5 most similar candidates. Based on the above we can see that the system 1 is far superior to the rest, with system 2 following, then system 3, then 4. After certain ranks, which are very high, system 3 outperforms system 2, but we are not concerned with such high ranks.



- **Compute the Rank-1 Recognition Rate.**

```python
def calcRank1(simMatrix):
    numQueries = simMatrix.shape[0]
    # Initialize count of Rank-1 successes
    rank1Successes = 0
    for i in range(numQueries):
        # Get the scores for the i-th query
        scores = simMatrix[i]
        # Determine if the genuine score is the highest
        if np.argmax(scores) == i:
            rank1Successes += 1
    # Calculate the Rank-1 Recognition Rate
    rank1RecogRate = rank1Successes/numQueries
    return rank1RecogRate
```

```
System 1 Rank-1 Recognition Rate: 0.991
System 2 Rank-1 Recognition Rate: 0.832
System 3 Rank-1 Recognition Rate: 0.031
System 4 Rank-1 Recognition Rate: 0.004
```

The rank 1 recognition rate is the percentage of times the correct identity is the one with the highest score (rank1). Here again we get the same results of superiority with descending order of systems 1, 2, 3, 4.

Q7: Evaluate different biometric systems

- **Propose a new metric not mentioned in this task to evaluate these four systems. This can be something you come up with yourself by adjusting/combining some of the existing metrics in this task or something you find online. Explain in what way your new method is more/less suited to evaluate these systems**

I would propose a weighted scheme of some sort that would incorporate metrics such as F1, AUC, a top1 rank rater and maybe a couple more in a linear way along with user experience satisfaction for a system, since each metric alone is not enough and, user experience in biomedical cases is very important.

$$\mathbf{Metric} = \mathbf{a} * \mathbf{f1} + \mathbf{b} * \mathbf{PRCAUC} + \mathbf{c} * \mathbf{rank1} + \mathbf{d} * \mathbf{UX}$$

Where a + b + c +d = 1

Biometrics Assignment 1 Report
Dimitrios Chatzakis r0977164

Q1: Now that we have some results, is the given similarity function a good metric to quantify a distance between two fingerprints? Is it reliable enough to incriminate a suspect? What are its limitations?

The MSS is highly sensitive to exact matches in value and position within the arrays. For fingerprint data, which typically involves complex patterns and features, small shifts or distortions can lead to large differences in MSE, even if the fingerprints come from the same person. Also, this measure does not consider the spatial relationships and unique features like minutiae points, ridges, and bifurcations that are critical in fingerprint analysis. It treats the data as a simple numeric array without any contextual processing.

Q2: Take a look at the enhanced images. Do you see any challenges that you can face working with these?

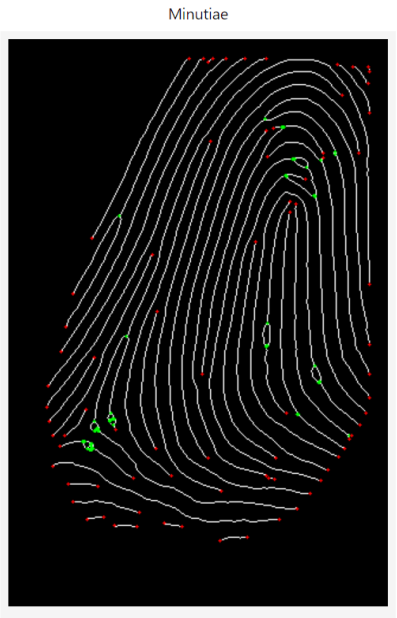
As we can see above, the fingerprints contain erosions of their minutiae, so the fingerprint loses its distinctiveness and reliability for identification.

Q3: Write your detection algorithm below. Keep in mind when you're inspecting the 8-neighborhood centered on a white pixel, the area also includes the pixel you're inspecting. Don't forget to update the criteria with this in mind! Explain in your report how your algorithm works and show some example(s).

The algorithm operates by first determining if the central pixel of a given region holds a value of 255, indicating it is a point of interest. Following this initial check, it evaluates the sum of the neighboring pixels' values. Based on this sum, the algorithm classifies and records the type of minutiae:

- If the sum of the neighbors is 1, the algorithm identifies this scenario as a ridge ending. It then stores this finding in the minutiae list as a tuple (x, y, True) where (x, y) represents the pixel coordinates, and the value True signifies a ridge ending.
- Conversely, if the sum is 3, it indicates a bifurcation point. The algorithm appends (x, y, False) to the minutiae list, with False denoting a bifurcation.

This method efficiently records the location and type of key features in the fingerprint, aiding in precise identification tasks.




```
# Avoid edge cases
for i in range(1, rows - 1):
    # Same as above
    for j in range(1, cols - 1):
        # If pixel is of the skeleton
        if skeleton[i, j] == 255:
            # Get its neighbors sum
            num_neighbors = int((temp[i-1,j-1]+temp[i-1,j]+temp[i-1,j+1]+
                                 temp[i,j-1]                +temp[i,j+1]+
                                 temp[i+1,j-1]+temp[i+1,j]+temp[i+1,j+1]))
            # if it has only one it is termination
            if num_neighbors == 255 * 1:
                # Create tuple with coords and True
                minutiae_list.append((j,i,True))
            # same for 3 and bifurcation
            elif num_neighbors == 255 * 3:
                # Else x,y,False
                minutiae_list.append((j,i,False))
```

Q4: Comment on the advatages of detecting the orientations of minutiae on top of just locations.

The orientation of a minutia provides critical directional information about the ridges in a fingerprint. By incorporating both the location and orientation of each minutia, fingerprint matching algorithms can achieve higher accuracy since this additional attribute ensures that the minutiae being compared not only exist at similar points but also align correctly in direction, reducing false matches and increasing the likelihood of correct identification. Also, advanced fingerprint matching algorithms could leverage such orientation data to create more detailed and precise models of the fingerprint ridges.

Q5: Are all the keypoint matches accurate? Are they expected to be? Explain why.

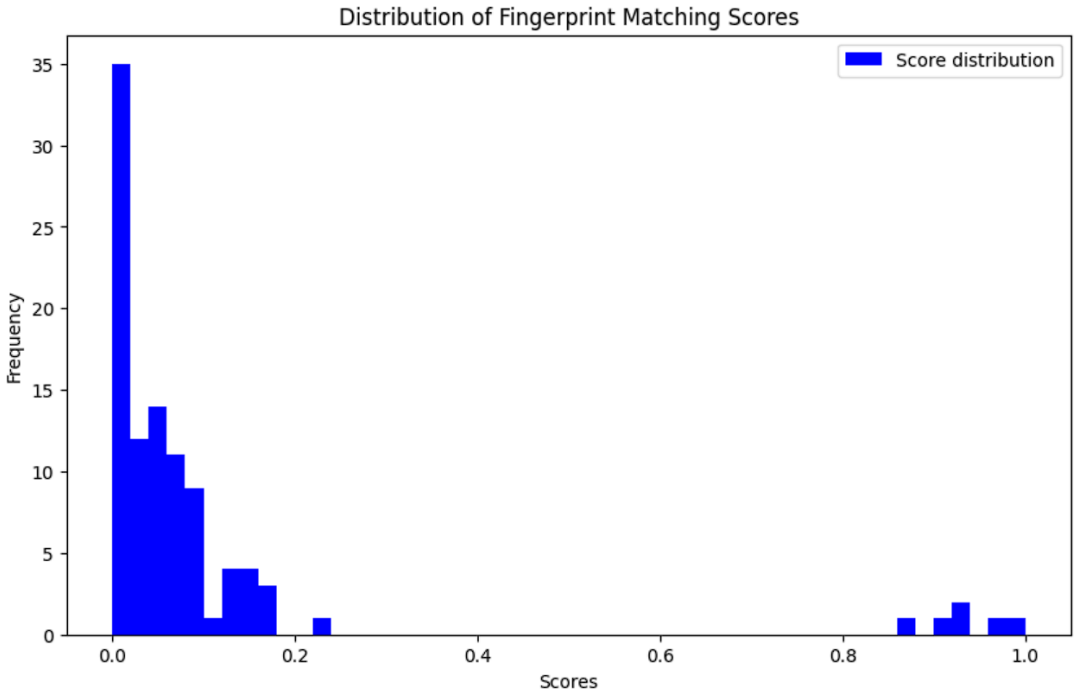
No, not all keypoints are accurate since the algorithm uses the cv2.estimateAffinePartial2D with RANSAC to find the best affine transform that aligns the matched keypoints. RANSAC helps to improve the robustness of the transformation estimation by iteratively considering subsets of matches and excluding outliers but despite that, the success of the affine transformation heavily relies on having a sufficient number of accurate matches. Inaccurate matches, even if few, could skew the estimated transformation if they are not identified as outliers by the algorithm. Also, descriptor-based matching fundamentally depends on the quality and distinctiveness of the descriptors. If they are not sufficiently distinctive, or if the keypoints are in texture-poor regions, mismatches are sure to occur. Environmental factors like lighting, viewpoint changes, occlusions, and noise could also affect their reliability, leading to potential mismatching or inaccuracies in general.

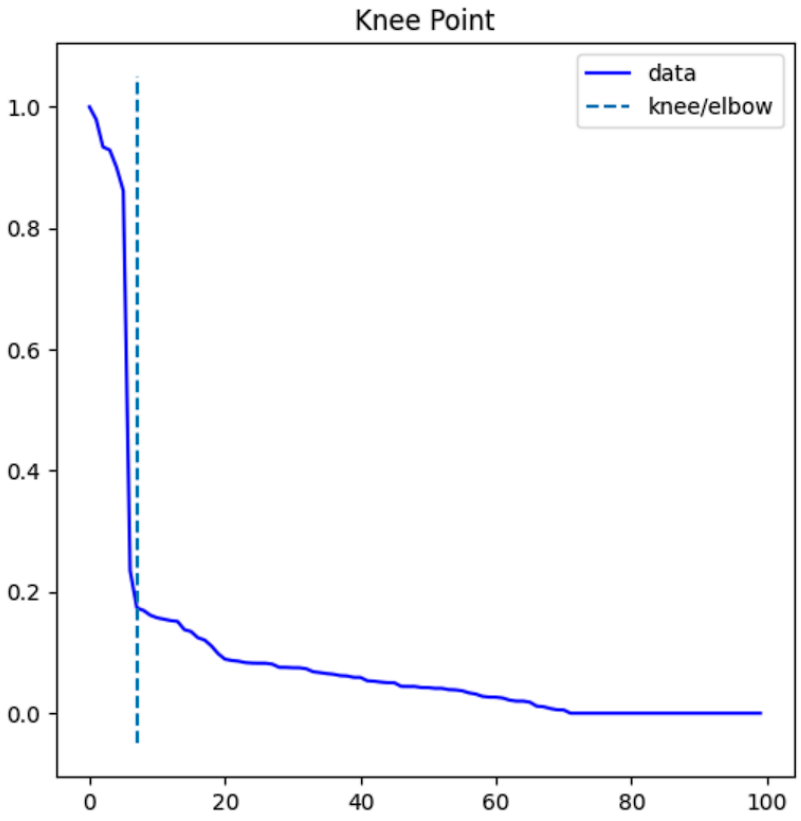
Q6: Choose a global feature similarity function, (e.g. you can start from euclidean distance between the reduced sets of KeyPoints and count the values above a threshold).

My global similarity function counts the number of matches between two images that have a distance less than or equal to 1 to have very few false positives.

Q7: Visualize the scores and determine a score threshold to discriminate the matching fingerprints. Explain how you determine the threshold.

The histogram with 50 bins along with the knee calculation are as follows:





The kneepoint is at 0.1736. As a threshold I would keep only the ones above above the last value before the huge depicted drop-off. The score function is normalized between 0 and 1.

Q8: Check out iris_perpetrator.png. Where do you see difficulties? What kind of similarity measures do you expect to work best?

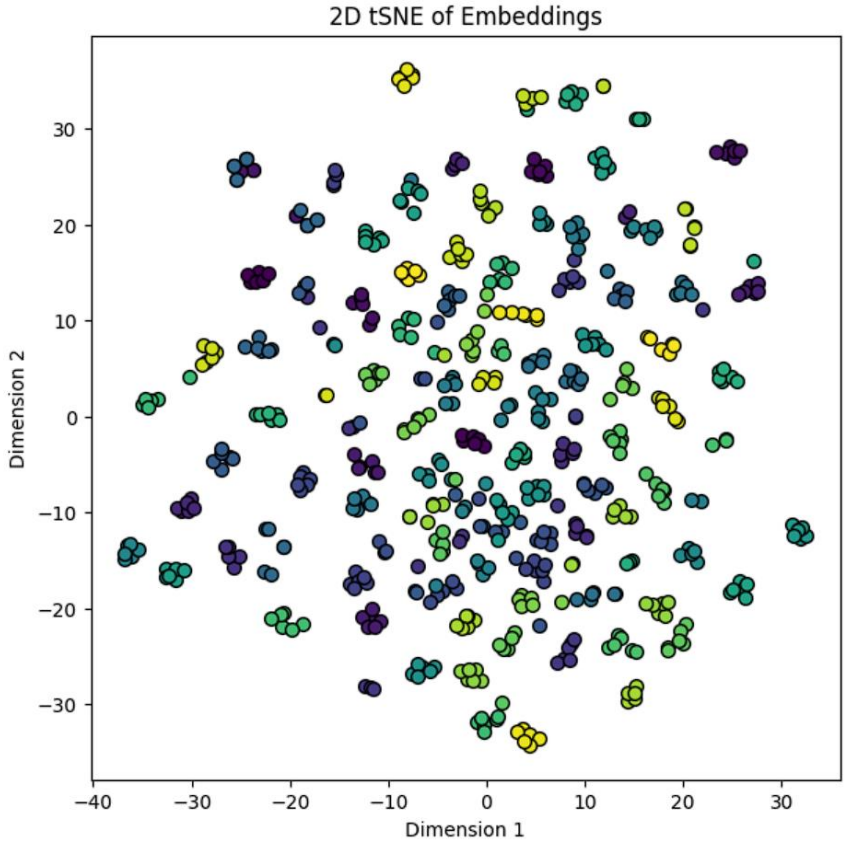
The eyelids, eyelashes could partially obscure the iris, leaving important features of the iris hidden and making it difficult to extract reliable features, however that is not the case in us. Variations in lighting conditions such as too much lightning can cause the pupil size to change, altering the visible area of the iris and thus affecting the patterns available for analysis as well as the segmentation maybe. Also, iris images captured from angles other than the default we have in the database can distort iris patterns due to the non-frontal presentation, which complicates feature matching.

Q9: Inspect the enhanced iris images. Do any of your difficulty predictions still hold? Do you see new ones?

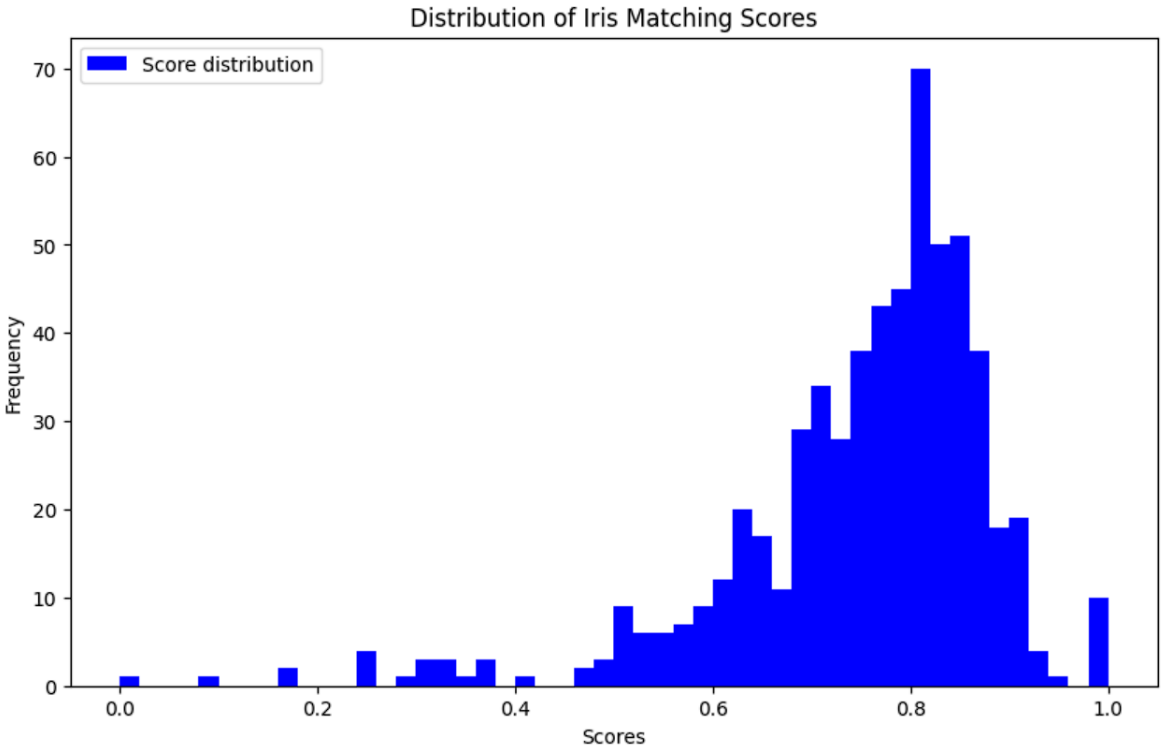
The produced images have two issues, some segmentations mask do not contain part of the eyelid, and several ones have eyelashes protruding into the iris. So yes, the ones I had predicted do hold for the images of the dataset.

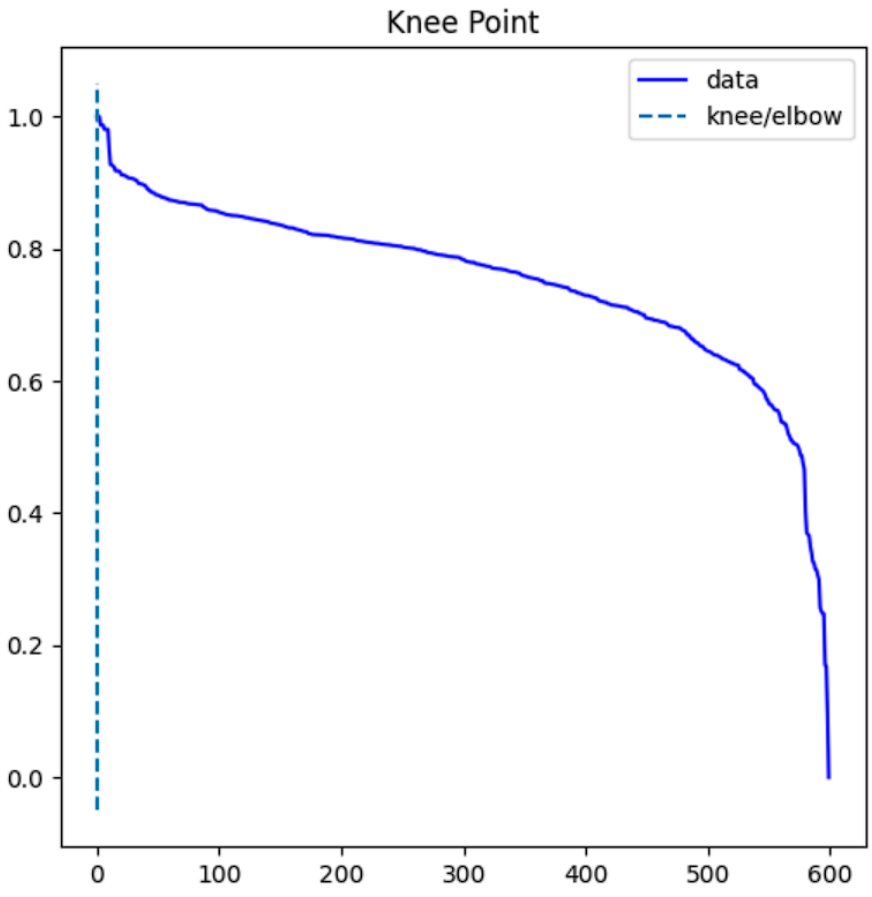
Q10: Visualize the embeddings in a lower-dimensional space (2D or 3D) using a dimensionality reduction method. Interpret the plot.

Here we visualize using tSNE and we see that in most cases the algorithm separates efficiently by grouping most labels together, despite having some false positives.



Q11: Ball's in your field. Enhance the perpetrator iris image, push it through the encoder you trained , and construct a similarity table just as you did with the fingerprints. Visualize your similarity scores and choose a threshold. Discuss results in your report.





In the visualization above the algorithm does not display a knee point however we can see that for the scores above around 0.9 we have a knee, so as a threshold I would choose that value. The score function is again normalized between 0 and 1.

Q12: Fuse your iris and fingerprint biometric system on the score level to make a prediction and solve the murder case! Use the metrics you implemented in the previous assignment to assess your system. Do you feel confident in your prediction? How do you fuse the scores? Why?

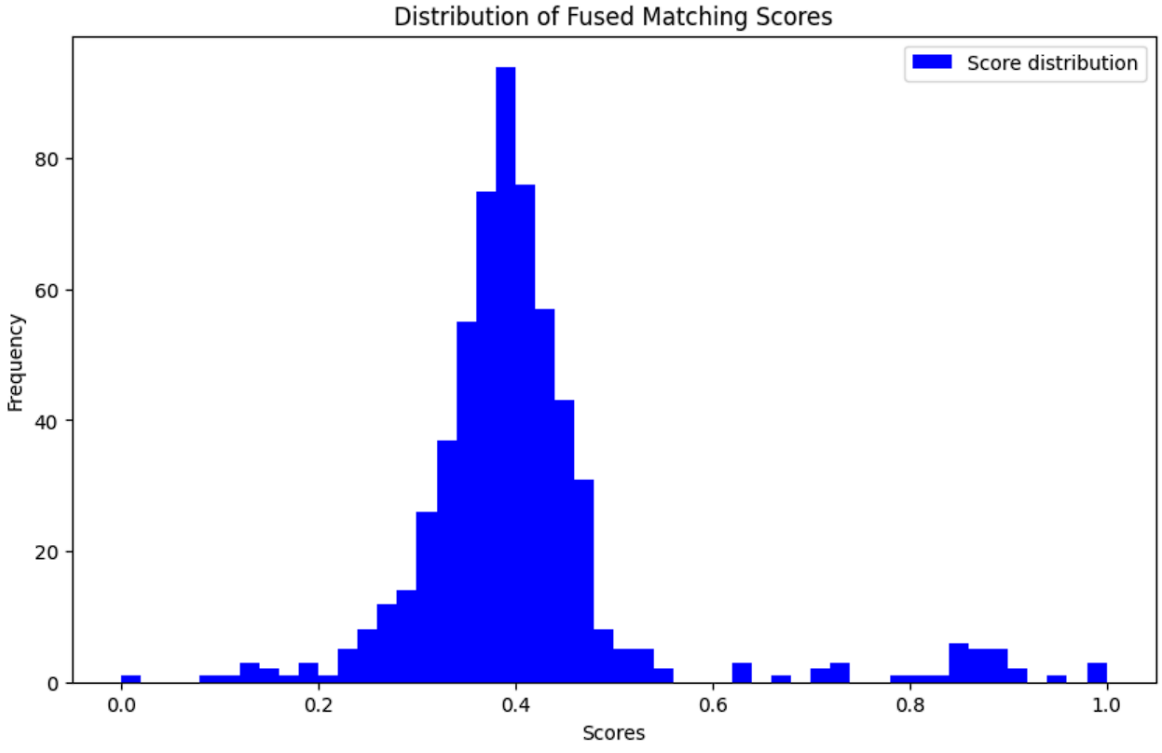
The proper way to fuse them would be I think the average between the two. Each fingerprint score for one label should be added to each of the 6 different scores for the iris and then averaged to find the perpetrator. This is done only on normalized data because we utilized different methods to calculate the dissimilarity. The top 5 suspects are:

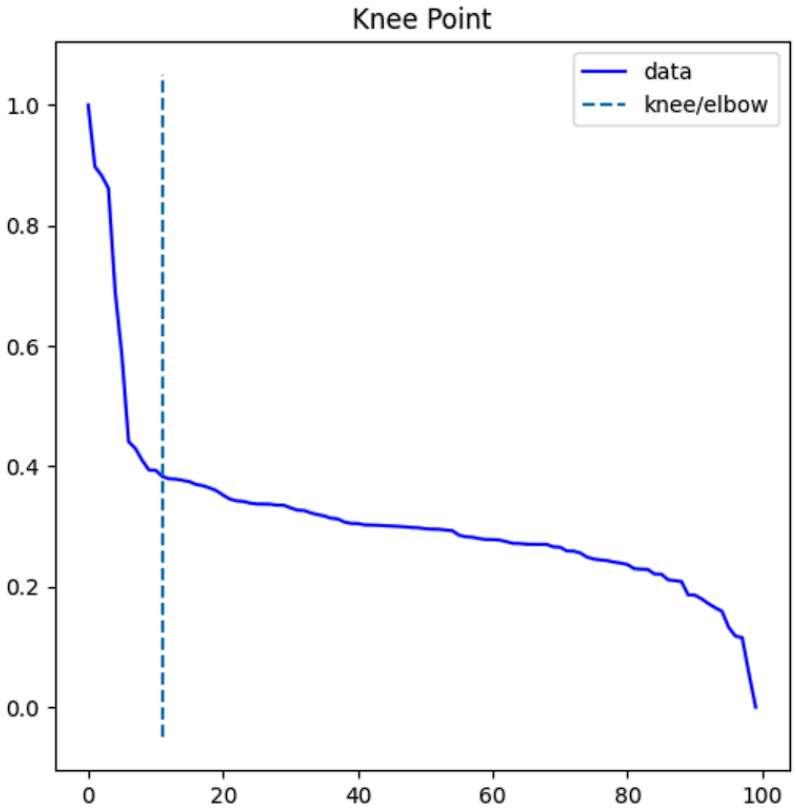
id	score_x	score_y	fused score
007	1.000000	0.923097	0.960215
072	0.900198	0.863233	0.877615
055	0.861569	0.878243	0.865396
019	0.933830	0.773543	0.848614
054	0.928896	0.511479	0.710487

Based on the results on the side I would argue that the suspect is safe to say that is the suspect with number 007.

Due to time complexity issues it would not be possible to compare using metric from the previous system because the calculations of the similarity matrices would be prohibitive. However, if I could I would use the combination of two metrics, the AUPRC and the rank1 rate.

I feel confident enough about my score but if I had time I would try out different thresholds in the distance part and add the angle differences. In addition to that, I would use also another metric in eyes, that is less sensitive to outliers, maybe the hamming distance. Below you will find the visualizations for the fused data.





Task B: In an authentication scenario, where the stakes are higher due to security reasons and there is a 1-to-1 comparison, the same method would be used maybe about fusion, but the person would be accepted only if the result was above a high predetermined threshold.

Task D: Implemented in a similar manner. Check the notebook included in the zip. The subtle point here is that due to the circular nature of the angles, you must keep the minimum between the absolute of angle differences and its subtraction from 360, e.g. think of difference in angles of 356 and 4 degress.

Task L: Due to time limitations again, I went for a quick and dirty solution. I resized the segmented iris images and collated them with the fingerprint ones, then parsed them through the model you provided. I saved the model and restarted the kernel several times due to crashing. After 50 epochs I reached a loss of 0.0000 and stopped the training. At that point I got the embeddings of training and validation set, the same ones you utilized, evaluated the model using an SVC with an rbf kernel, after parsing the embeddings through a gaussian scaler that was fitted in the training data. I reached an accuracy of 97% and an f1 score of 96%. However, when parsing the case in point, the predicted label was 71 with the others that were not at the top of the iris or fingerprint similarity score. Therefore, I think the model was not successful, probably due to its dimensionality in the output embeddings and the input image, which due to being resized lost a lot of information on two images.

Dimitrios Chatzakis r0977164
Biometrics Assignment 3

Q1: Compute distance-based pair-wise matching scores.

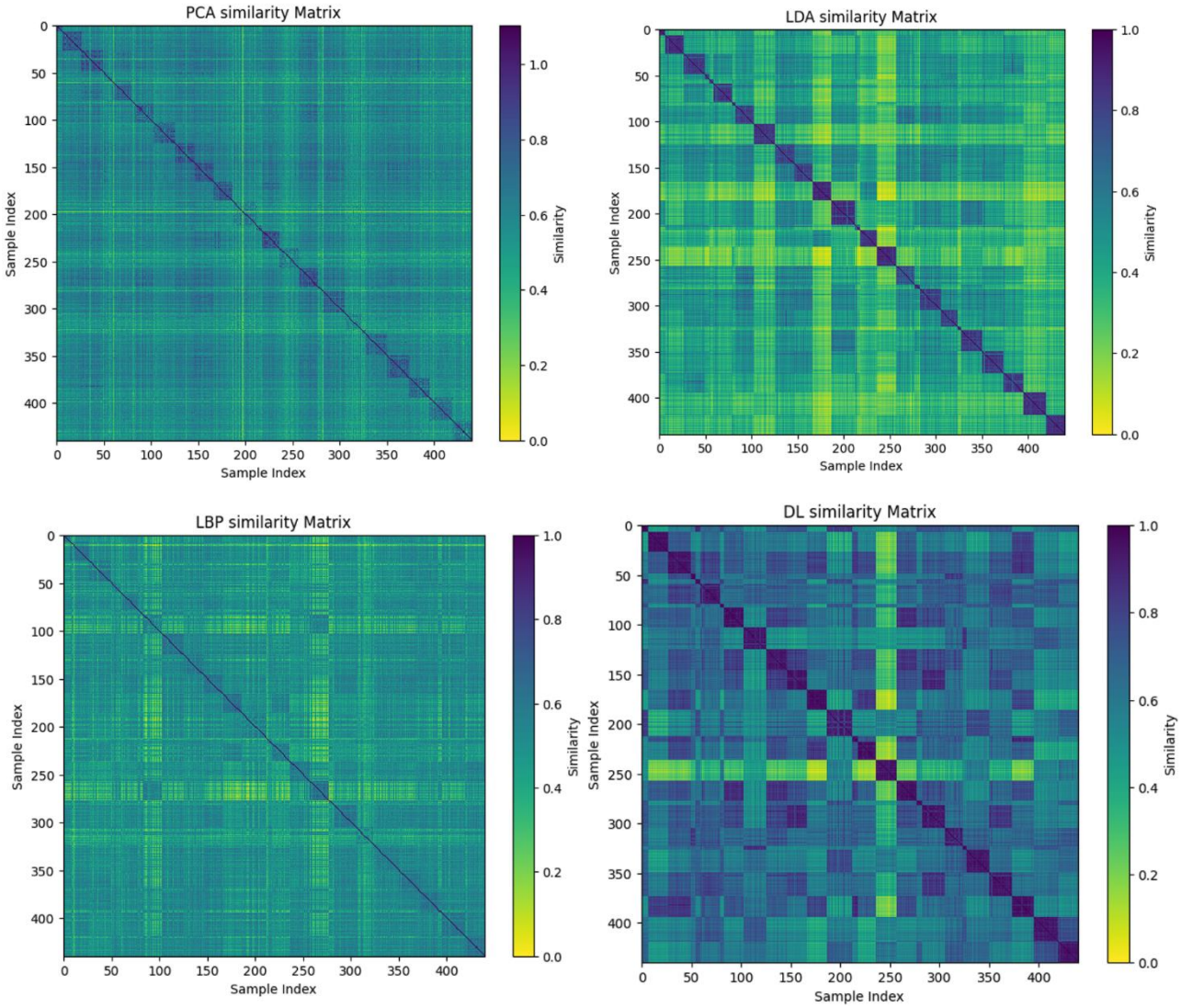


Figure 1 Similarity Matrices for Different Methods

Each diagonal square consists of photos from the same person and has thus higher similarity. They have higher contrast in well performing systems and less in bad performing ones. From the above we can see some initial intuitive results, namely that LDA will be one of the best techniques and LBP probably the worst.

Q2: Plot genuine and impostor scores. When comparing the different feature extractions/facial representations, discuss the difference in the overlap between genuine and imposter scores.

From the genuine and impostor scores we can see some more initial results (Figure 2):

***PCA*:** Large intra-class distance, medium inter-class, large overlap

***LDA*:** Large impostor intra-class distance, medium genuine, large inter-class distance, very small overlap.

***LBP*:** Medium impostor and genuine intra-class distance, small inter-class distance, very large overlap.

***DL*:** Large impostor intra-class distance, small genuine, large inter-class distance, small overlap.
Given that Large inter, small intra, small overlap are the better results I would say LDA, DL, PCA, LBP in descending order of robustness, but let's continue the research.

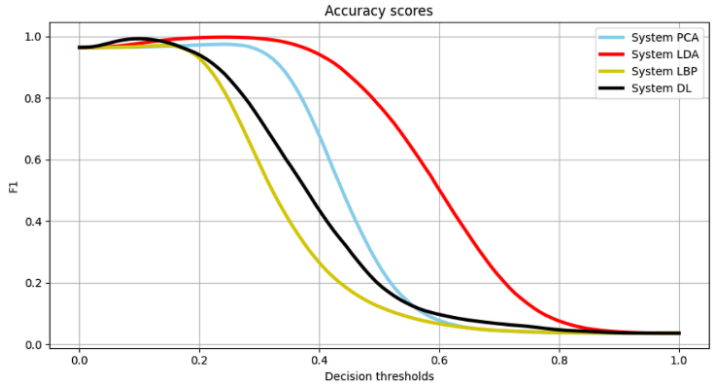


Figure 2 Accuracy distributions for each method

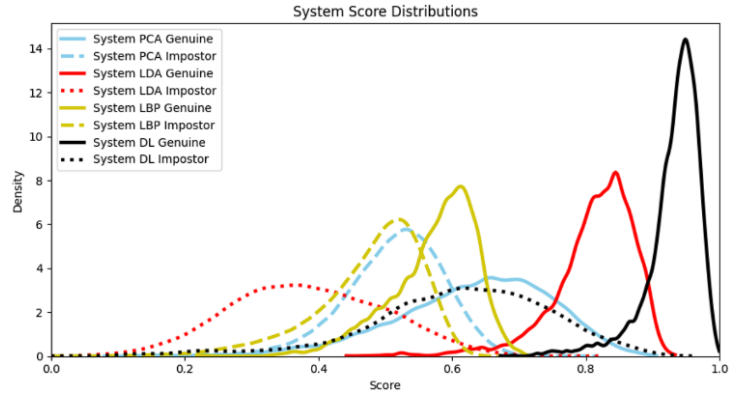


Figure 3 Score distribution for each methods

Q3: Compute F1 and accuracy scores for variable (and optimal) thresholds. Calculate and plot F1 and accuracy scores for a range (min, max) of thresholds. Determine some optimal threshold (look up in classification literature). Justify your answer.

For each label, we have multiple images and I calculate the metrics in an unweighted manner, calculating the accuracy and f1 for each person and then calculating the average for the 26 different people, always based on the distances not the similarities. That is done based on the assumption that a system should classify each person well, with each person having the same importance, irrespective of the number of images.

The only interesting point in accuracy, since it is a bad metric for biometrics, given it checks only for true positives and true negatives, and is thus unsuitable for the unbalanced datasets of biometrics, is that there is an upward stroke at the start of the curves. This is since for each label we have many images, and the comparison is based on distances, at the start we have a lot of True Negatives and False Negatives without True Positives. As the threshold slowly progresses, many False Negatives turn into True Positives, faster than True Negatives turn to False Positives. Then the situation is reversed and continues the regular trend stopping at the point where there are the few only true positives and all the other labels are False Positives. Next are the F1 scores (Figure 4), here we can see the superiority of LDA relative to other methods for identification tasks, followed by DL, PCA and LBP. As previously discussed in Assignment 1 in verification, where we have a 1-to-n comparisons and usually the cost of making a mistake is not usually that big (precision and recall have similar weight), as well as there being imbalanced datasets with many True Negatives (which the f1 doesn't take into account) the f1 is a robust metric.

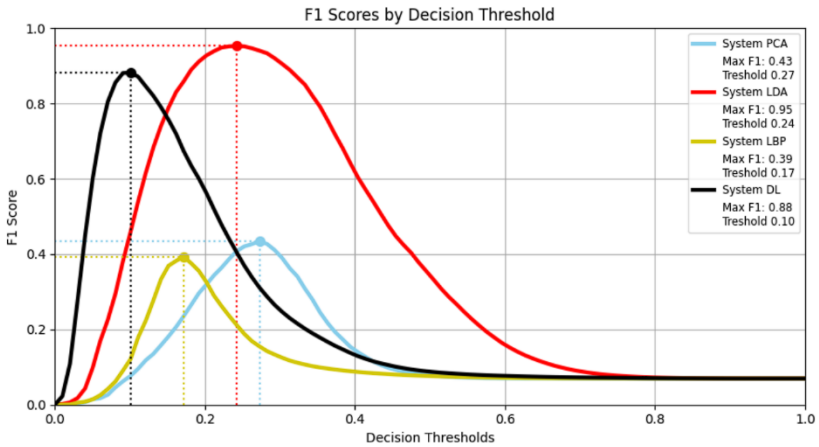


Figure 4 F1 scores for each method

Q4: Perform a full-on verification assessment based on the scores obtained. Interpret the results.

First, I calculated the EER for each method.

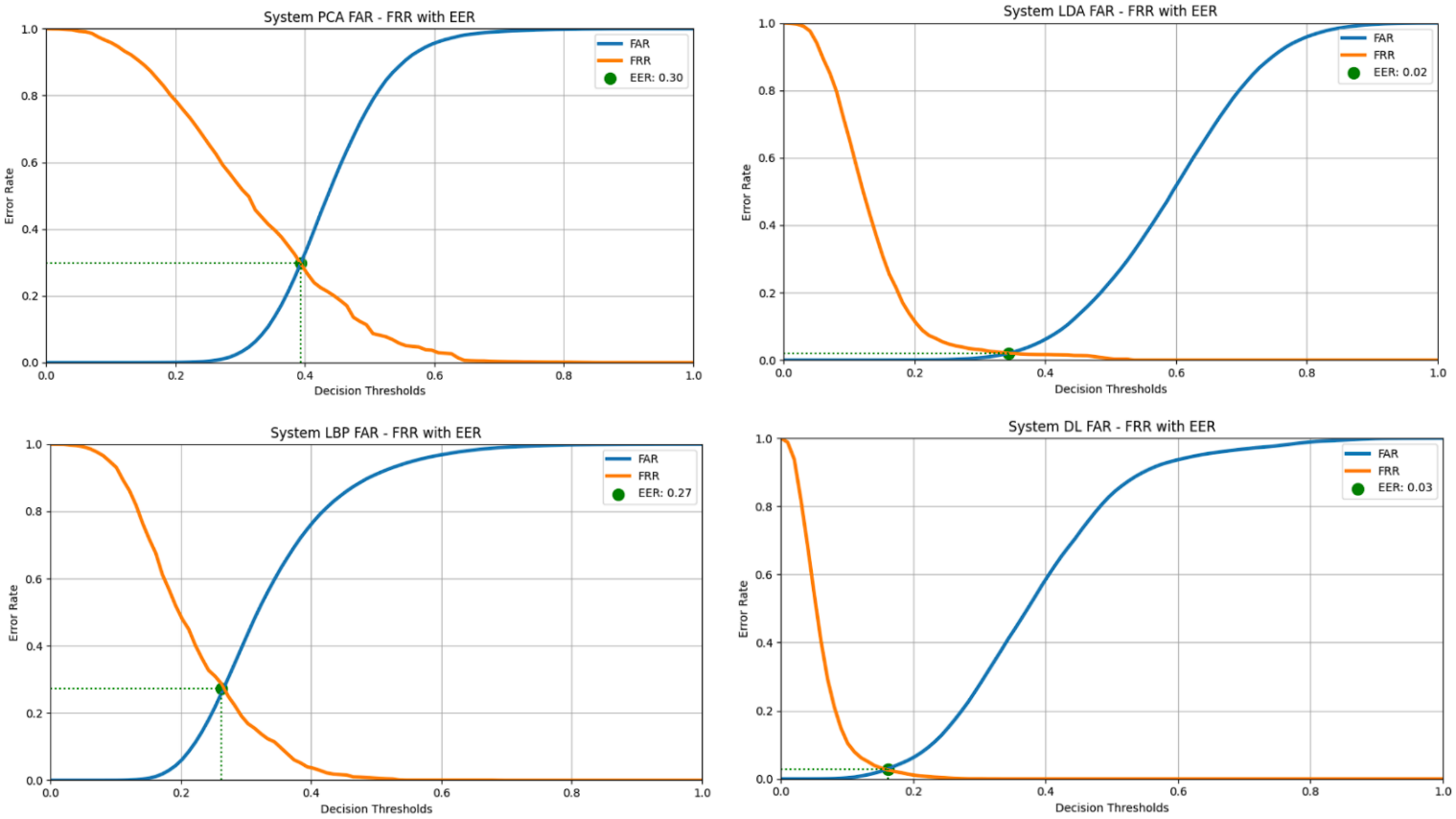


Figure 5 EER for each method

EER represents the point where the system's sensitivity (avoiding rejecting legitimate users) and specificity (avoiding accepting impostors) are balanced. The lower EER (better system) is in LDA, closely followed by DL, then LBP, closely followed by PCA. Based on the PRC Figure (Figure 6), we can see that, the best AUC is for DL, LDA, PCA and lastly, LBP. The average precision by far the highest for the LDA, followed by PCA, then DL and lastly, LBP. Given that the precision in biometrics is usually more important and that AUCs are pretty similar, along with the fact that for high thresholds we have almost a perfect system in LDA upper right corner of graph, I would pick the LDA.

Q5: Validate the systems in an identification scenario. Calculating the CMC for each class (Figure 7), we can see that the CMC of LDA and DL are close, followed by LBP and PCA. Given that the above Figure has accounted for class imbalances in the images, I would argue that for verification scenarios the best methodology is the DL since its curve is always above the rest and has higher rank1.

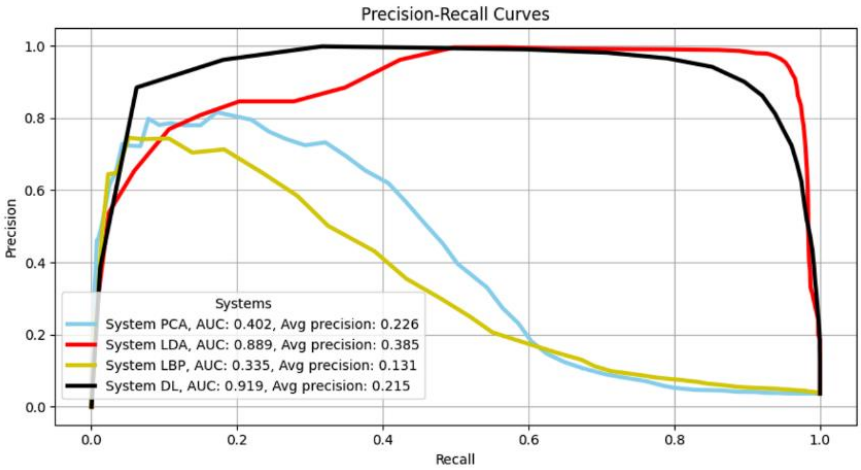


Figure 6 PRC Curves for each method

Q6: Fine-tune your model for feature extraction.

For all the methods I used the F1 and the CMC's rank-1 rate, the f1 since for every system and application there is an optimal f1, and is more important than the PRC which is more suitable for generalization of the utility of a system. Rank1 shows how often the correct person is identified first, and is the more robust metric than the CMC given that usually a wrong identification is costly. An interesting metric to implement is the max f1 for each system for different weights of precision and recall.

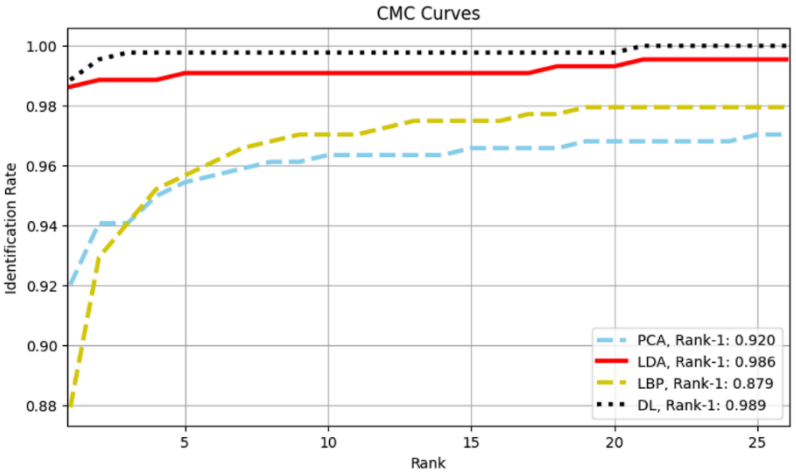


Figure 7 CMC plot for all methods

PCA: For similar reasons as before we can see (Figure 8) for the verification scenario the best method is very clearly decided. The f1 score is better clearly for the case of 20 components with an AUC that is better. For the identification case I would pick the System 50 since it has the highest rank-1 rate.

LDA: For LDA, which is much better overall for reasons I will comment in the end of this question, the LDA with 25 number of components (the original) seems to be the best. It has the highest F1. In the identification case, I would pick the 25 components case as well since it has higher rank-3 rate as can be seen from the graph (although not computed) having the same rank-1 as the system 15.

LBP: The LBP with a radius of 4 yields superior results in F1 and rank-1 metrics.

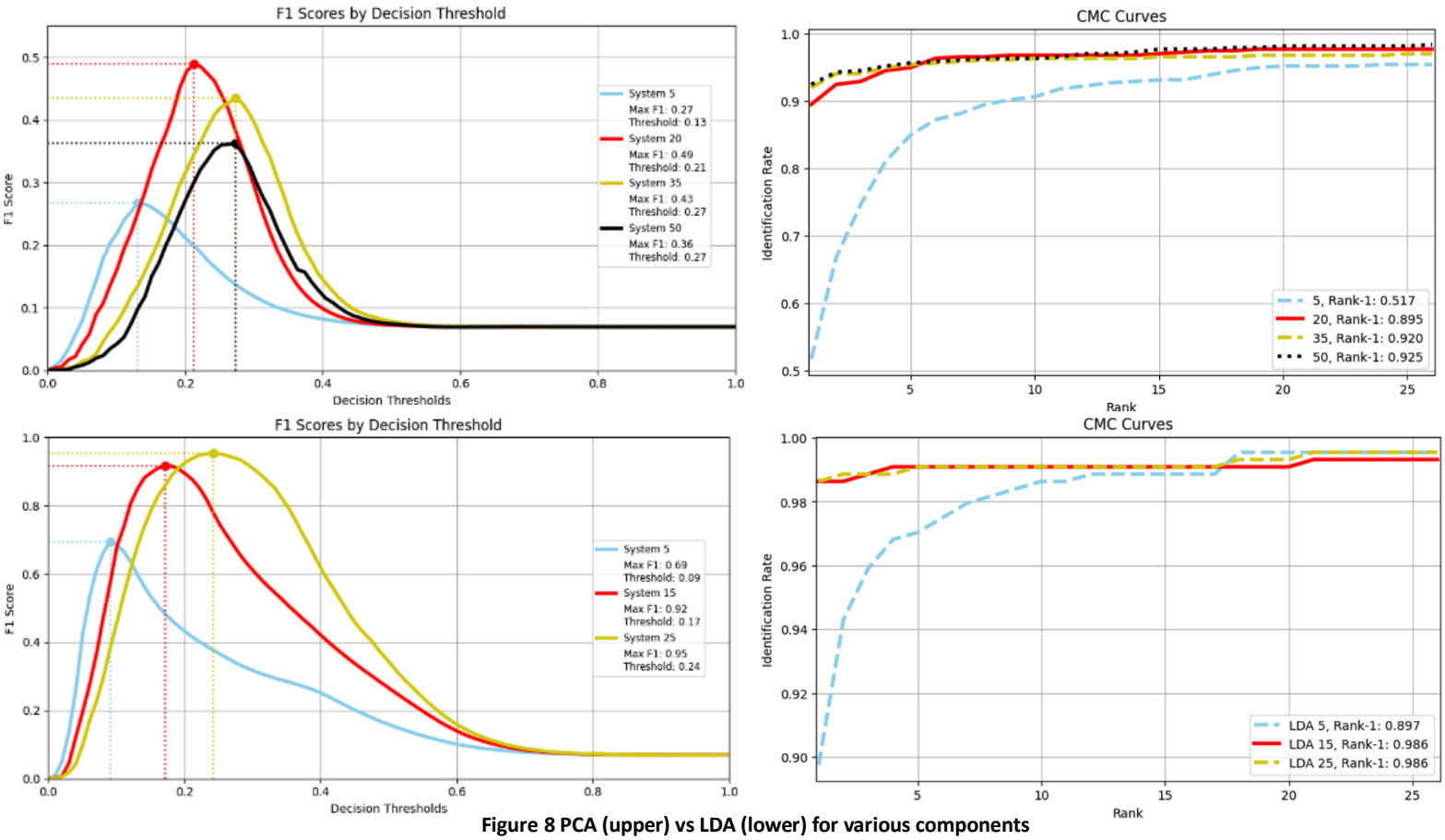


Figure 8 PCA (upper) vs LDA (lower) for various components

DL: The best model for verification is the one with 64 neurons in my opinion, since it has the highest F1. For identification, I would pick the original model since its CMC is equal or higher than the rest.

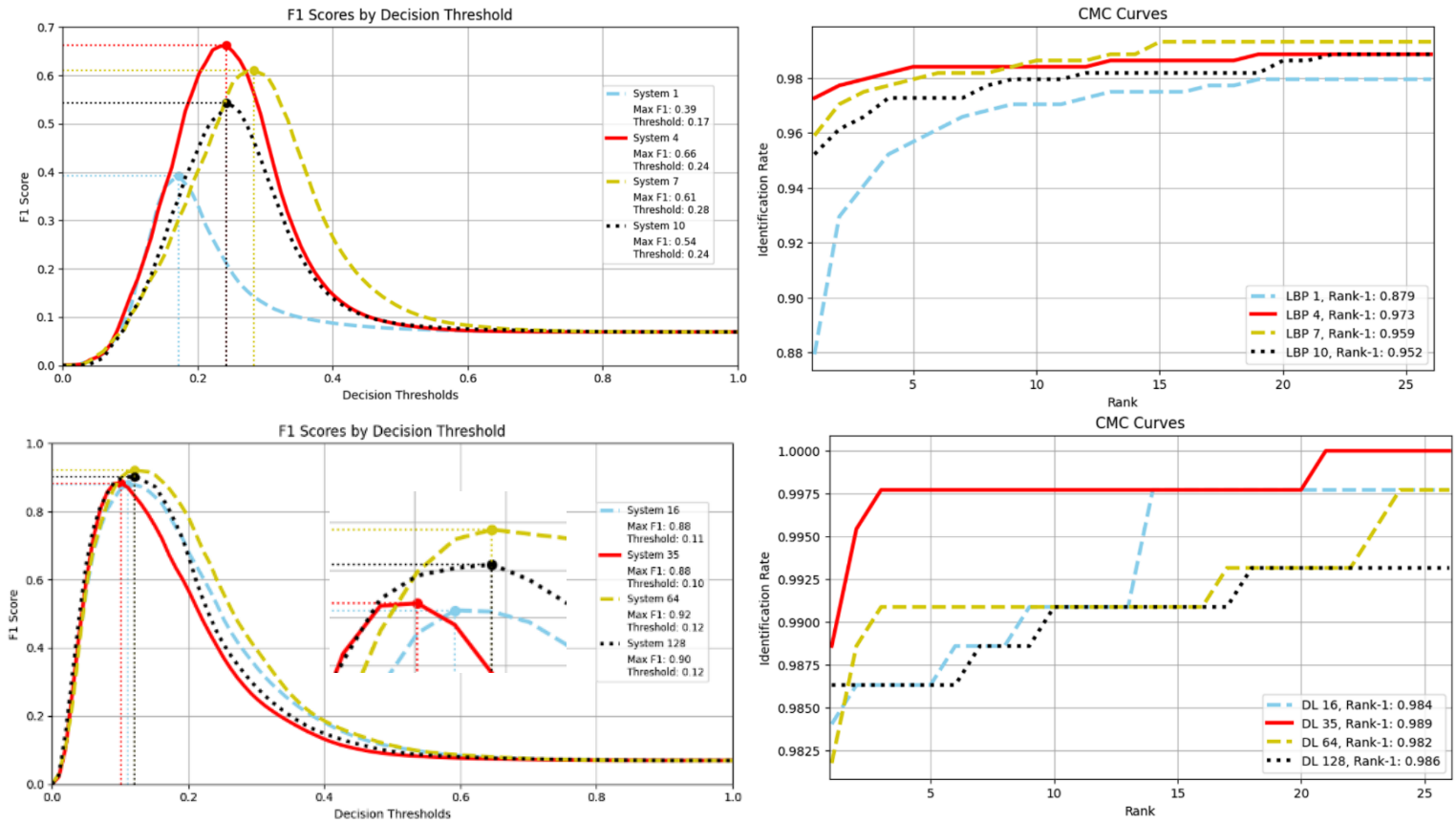


Figure 9 LBP (upper) vs DL (lower)

Q7: Compare the 4 feature extraction models (LBP, PCA, LDA, DL using Siamese Networks)

Given Figure 10, the best model is the LDA for the verification and the DL for identification.

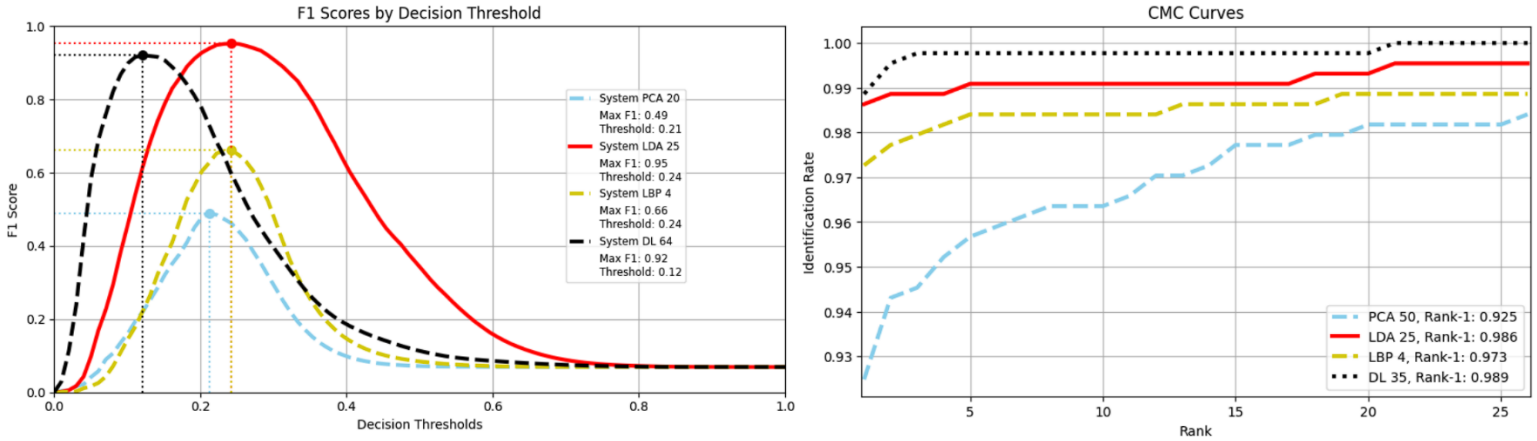


Figure 10 Comparison of methods with optimal parameters.

The overall results are somewhat expected. LBP is a local texture extractor, robust to lighting changes that are common in faces. PCA is illumination sensitive due to capturing global features and doesn't capture local ones while also being class agnostic. Then for LDA, it gets information about the

undelying classes and tries to maximize inter-class variance, practically what the biometrics system want to achieve. Retaining more components, I would expect The results for the Deep Learning architectures are to be expected, however, I would expect there would be a need for more complex models. All in all, I would expect to have the DL achieve overall better results, followed by LDA, then LBP followed by PCA, which happens as a general trend. However, in a verification scenario the LDA seems to reign supreme while in the authentication the DL is more robust.

Task V: Implement a new model.

I wanted to try out how the more complex models would perform so I implemented a variation of the original model with a twist, I added squeeze and excitation attention (SEA), and then also used a ViT to gauge how much better the models would be. I expected the CNN with SEA to be a bit better but the ViT to be a bit of an overkill and overfit since Transformers that have a lot of parameters and are notorious for overfitting in small datasets. The results are shown in Figure 11 below.

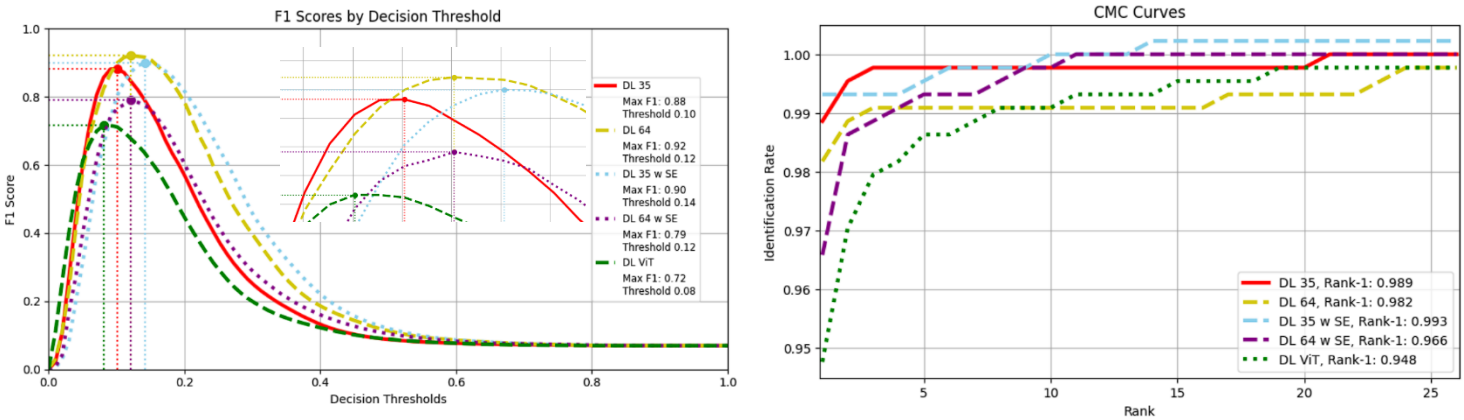


Figure 11 Comparison of custom models

We can observe that the best model for verification is still the default model with 64 embeddings with the SEA model with 35 embeddings being a close second. The SEA with 64 embeddings as well as the ViT fails to generalize well and thus have lower F1s. In the Identification scenario, I would pick the SEA with 35 embeddings for its higher score. Again, we see the other SEA having subpar results as well as the ViT for the above aforementioned reasons.

GENERAL REMARK

It is natural, although undesirable that different systems are better for different tasks. While a DL model could be better for identification with more components, since it needs a 1-to-1 comparison and can handle overfitting better, the imbalanced nature of verification having 1-to-n comparisons may not work due to that same model overfitting. The same behavior could be seen also in PCA. LDA seems very robust in both cases.

Appendix: Custom models

Squeeze and Excitation Attention

The primary objective of the SEA block is to enable the network to perform dynamic channel-wise feature rescaling. This allows the network to learn to emphasize informative features and inhibit less useful ones across channel dimensions. First, a transformation occurs:

$$F_{tr} = u_t * X = \sum_{s=1}^{C'} u_t^s * x^s$$

where $F_{tr}: X \rightarrow U$ is a convolution transformation from $X \in \mathbb{R}^{W' \times H' \times C'}$ to $U \in \mathbb{R}^{W \times H \times C}$.

The squeeze phase aggregates spatial features into a channel descriptor. This is achieved by using global average pooling to compress each channel of the feature maps into one scalar. Thus, the network captures the global distributions of features on each channel

$$F_{sq}(u_t) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} u_t(i,j)$$

where u_t is the input at timestep t, and W, H are spatial dimensions.

Then, the excitation step aims to capture channel-wise dependencies. This is typically done through a simple gating mechanism involving a sigmoid activation and a ReLU activation through two layers:

$$F_{exc}(x, W) = \sigma\big(g(z, W)\big) = \sigma(W_2 \delta\big(W_1 F_{sq}\big))$$

where $W_2, W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and r is the reduction ratio. The first layer reduces dimensionality by a factor of r to limit model complexity and aid in learning channel-wise dependencies. The second layer then scales it back to the original number of channels:

$$F_{scale}(u_t, F_{exc}) = F_{exc} u_t$$

Transformers

Self-Attention

The attention of transformers, self-attention, maps the input to Query, Key, and Value vectors. The essence of self-attention lies in calculating the relevance or importance of each input in a sequence concerning every other input in a sequence. The following operations are then performed: Calculation of the dot product between the Queries and the Keys and scaled by $\sqrt{d_k}$, their dimensionality. This scaling factor ensures that the dot products remain stable and are not overwhelmed by the dimensionality of the vectors. Then, after applying any masking one might want to utilize, one applies a softmax activation function to the result to calculate the weights to be applied to the Values:

$$SelfAttention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

resulting in attention weights that represent the importance of each input in a sequence (here pixels).

This architecture was inspired by how databases work in information retrieval. The query vector represents the element of interest or the context that you want to obtain information about, The key vectors are used to compute how relevant each element in the input sequence is to the query and the value is the information returned based on the previous two. I found this short that sums it up very well: “The Query seeks relevant information, the Key identifies it, and Value provides it.”

Multi-Head Attention

Multi-Head Attention extends the self-attention mechanism by employing multiple sets of Query, Key, and Value vectors, known as "heads." Each head operates independently and learns to capture different aspects of the input sequence. This parallel processing enables the model to attend to diverse patterns and dependencies within the data simultaneously in fashion similar to how convolutional layers capture different aspects of the input signal. In this setup, the self-attention operation is

performed h times, where h represents the number of heads utilized. The outputs of all heads are then concatenated and linearly transformed to obtain the final output.

Transformers

Transformers are the state-of-the-art architecture that leverages self-attention mechanisms to capture dependencies across input sequences more effectively. At the heart of the Transformer model lies the self-attention mechanism discussed above. This enables the model to understand the context and relationships between input in a more sophisticated manner and larger spatial context than previous models. The process starts by embedding each word in the input sequence into a high-dimensional vector space. These embeddings are learned during the training process and capture semantic information about the words. Additionally, positional encodings are added to the embeddings to convey information about the order of words in the input.

Next, the self-attention mechanism computes attention scores between all pairs of words in the input. These attention scores determine how much each element in the input should attend to other words in the sequence during processing. Importantly, the attention scores are calculated dynamically for each input element, allowing the model to adapt its focus based on the specific context. After computing attention scores, the model generates weighted representations of each word by taking a weighted sum of the embeddings of all words, where the weights are determined by the attention scores. This step aggregates information from all elements in the input, with each element attending more to elements that are semantically relevant in the given context. To capture diverse aspects of the input sequence, the self-attention mechanism is applied multiple times in parallel, each with its own set of parameters through the use of the multi-head attention mechanism described above and their outputs are concatenated to produce the final output of the self-attention layer.

The output of the self-attention layer is then passed through a series of fully connected feed-forward neural networks, which further process the information captured by the self-attention mechanism. Finally, the output of the feed-forward networks is fed into the next layer of the Transformer model or used as the final output, depending on the task being performed.

ViT

Initially, the transformers were utilized for words and then time-series. The same principles apply in the Vision Transformer but initially, ViT splits the original image into non-overlapping patches of a set size and then processes each patch to create embeddings.