Griffin Danner-Doran
Naif Alkaltham
Amelie Ferrell
Ife Adetunji
Roneet Arora

CSEC 472 - Group 2 Lab 4 Report

Github: https://github.com/gtd6864/lab4Biometric

Set Up:

For our methods, we used 3 different ML classifiers, those being KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and Random Forest. Each algorithm is trained using the 1500 training fingerprints. The 1500 original pairs as well as 4500 pairs of non-match fingerprints are used to fit each classifier. Once trained, the classifiers output a probability of a match for any provided features.

3 Methods:

In order to find the scenario where FAR and FRR are closest, each algorithm was tested with varying probability acceptance thresholds. In these tests, thresholds were tested from (.01 to .99) 1% to 99%, where the probability of a match output by the classifier had to be greater than or equal to the threshold value to be considered a match. This testing was done in the file **3methodstesting.py**, and These tests resulted in the EER values shown in the following table. The threshold values where each EER was observed were 31%, 13%, and 28% for KNN, SVM, and RF respectively.

Next, to calculate the max, min, and avg for FAR and FRR with each method, we tested each method by running **KNN.py**, **SVM.py**, and **Random Forest.py** 10 times each. In each of these tests, a subset of 100 values was chosen from the possible set of 500 test fingerprint pairs, which contain matches as well as non-matches, and was used to generate a set of FAR and FRR values for that testing run. The max, min, and average values resulting from those 10 runs for each method are shown below.

Note: For EER calculations, FAR and FRR are considered equal if they are within 7% (.07) of each other. The final EER shown in the table is the average of the FAR and FRR where such similarities occurred. Additionally, while the KNN and RF methods tended to output consistent results, the SVM had some variance between training runs, so the lowest EER observed is put here. The threshold value is the same for all variants, so this does not impact the accuracy of the 3 individual methods. Also, **3methodstesting.py** only returns accurate EER values if they occur within the acceptable threshold range on any given run, which may not occur on all runs.

Method	FRR avg	FRR min	FRR max	FAR avg	FAR min	FAR max	EER
KNN	44.4%	39.7%	53.8%	43.4%	26.4%	53.6%	46.7%
SVM	52.2%	42%	57.7%	54.6%	46%	67.4%	53.6%
Random Forest	44.9%	35%	53.2%	47.2%	41.2%	52.7%	47.7%
Hybrid	49.6%	40.8%	57.1%	43.7%	36.4%	64.7%	48%

Hypothesis:

While the lab instructions suggest a simple majority decision for the hybrid system, we instead chose to do a probability average solution where we combine the matching probabilities of each ML classifier, take the average, and then use that average probability to make the final matching decision. Furthermore, we added some weights to the different methods based on their accuracy. The KNN and RF methods were more accurate, so we gave them a weight of 4, while the SVM was less accurate, so we gave it a weight of 2. Then, for the final probability, we divide by 10. We predict this will be more effective than just doing a majority decision vote because it will include some of the variances between the algorithms in the final decision rather than reducing each vote to a yes or no and prioritizes the "opinion" of the more accurate algorithms.

For example, in a new test, KNN predicts a match with 80%, SVM predicts 60% and RF predicts 30% with a weight of (4-KNN, 4-RF, 2-SVM). In the majority decision approach, this would be interpreted as two votes for a match (KNN and SVM both above the 50% threshold), and one against (RF), leading to registering a match. Also, in our example, using the weights 4 for KNN and RF, and 2 for SVM, if KNN predicts an 80% match, SVM predicts 60%, and RF predicts 30%, the weighted average would be calculated as follows: (4×80+4×30+2×60)/10=52% (4×80+4×30+2×60)/10=52%. This method contrasts with a majority decision approach, where two votes for a match (KNN and SVM above 50%) and one against (RF) would result in a match. While an extreme example, this shows how this hybrid method would better include the predictions of each individual method. This in turn allows the KNN and RF systems, which tend to have a lower FAR and FRR than the SVM, to have a little more weight in decision-making since their probabilities will tend to be slightly more accurate than the SVM method.

As a result of this, our hypothesis is that our hybrid system will have a moderately lower EER than the current average of the 3 existing systems (49.3%), but it will not be quite as low as the EER of the KNN system or RF system.
Hybrid system:

Similar to the individual methods, we tested the hybrid solution with probabilities 1% to 99%, using this value as the threshold of acceptance for the combined average probability. This testing was done using **hybridtesting.py**. The EER was also taken in a similar manner to the other methods but a 2% difference was allowed due to the better granularity of the hybrid solution. These tests resulted in the EER shown in the above table. The threshold value had multiple potential options, but the best combined EER tended to occur at 27%.

Similar to each of the 3 individual methods, the hybrid method was tested by running **hybrid.py** 10 times. As with before, each run uses a random subset of the 500 testing fingerprints to test the method.

Note: As with the previous calculations, the SVM method introduced some variance into the hybrid system results, with the best observed case shown in the table for EER. Also, like the 3 method tester, **hybridtesting.py** does not always return an EER value.

Results:

Overall, our hybrid solution was successful in reducing the EER of the system relative to the other 3 methods as it had an EER of 48% compared to the average EER of 49.3%. It also had a surprisingly low avg FAR of 43.7% across the 10 cases we tested, but it is balanced out by the relatively high FRR of 49.6%. Additionally, as we expected, the EER of the hybrid solution was not as low as the KNN method, but was nearly as low as the RF method. Since the KNN method already had a lower FAR and FRR than the other 2 methods, the inclusion of the higher EER SVM method meant that the hybrid solution was unlikely to get as low as the KNN method, even with the SVM method having a lower weight.

As a result of these observations, our hypothesis was proven correct, as the EER of the hybrid solution was below the average EER of the 3 other methods, but still above that of the KNN method and the RF method.

The most surprising thing about this lab was how ineffective our basic image-processing methods were. We thought that the image computation method we did would be more effective, but it only had an accuracy slightly better than just guessing. Part of that was due to our relatively small training sizes, but if we had used a more complex method that was able to analyze the type and location of each feature, we would have had a much better EER. We discussed in class how fingerprint extraction is pretty simple, especially compared to other biometrics like iris and face recognition, but actually doing it made us appreciate how complex even these simple features are to recognize. This seems very different from something like iris recognition especially, where a lot of work goes into localizing + normalizing the iris and extracting features from it, but the final comparison is a super easy hamming distance calculation that can easily be done by hand. Overall, I think we gained a better understanding of how difficult the problem of biometric recognition is.

Future work:

In future enhancements, we plan to explore more feature extraction methods for our system. Currently, we just do simple image processing rather than doing a more complex minutiae extraction algorithm involving identifying specific minutiae locations. Alternatively, we could also test additional classifiers to further reduce the EER of our hybrid system. Particularly, we could replace weaker classifiers like the SVM to improve our EER. Finally, we only tested 1 match and 3 mismatches for every one of the 1500 training fingerprints to save time and computational resources. Adding more mismatches to the training data will allow our classifiers to become better at recognizing incorrect pairs, which can bring down our FAR significantly. Doing so will also reduce our EER. Additionally, we plan to enhance the training of our system with a more extensive dataset. This increase in training data volume is expected to further refine the model's accuracy and efficiency.

Further questions we have:

Following from the above paragraph, are there other classifiers that better suited for this kind of image matching that we could add to our solution to bring the EER down any further?
On a more general note, do real-life solutions tend to be hybrid or use a single method of feature extraction?