Baré Alexandre (r0912072)

Assignment 2 : Fingerprint and Iris Based Identification

Biometrics System Concepts

Katholiek Universiteit Leuven
Faculty of Engineering Science
Academic Year 2021-2022

1 Q1

The similarity function

```
mss = lambda x,y: 1/(1+np.square(x-y).mean())
```

does not seem appropriate.

As can be observed in figure 1, it does not discriminate enough the fingerprints in such a way that we could incriminate a suspect.

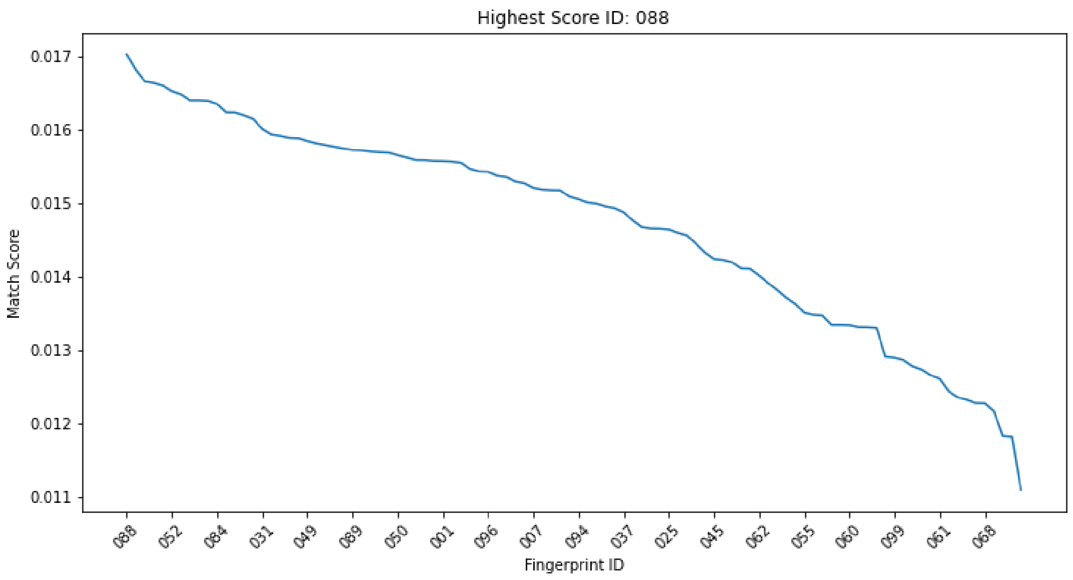


FIGURE 1 – Match Scores for Different Fingerprints

Relying on a squared error tend to over-penalize outliers and is thus very sensitive to noisy fingerprint images. Noise can come from poor capturing conditions (eg : dry, wet or dirty fingers, scars, ...). We notice for instance, the presence of discontinuities in the ridges and all ridges are not distinguishable from their neighbours. No preprocessing steps are applied to enhance and segment the fingerprints. Furthermore, comparing plane fingerprint images with the current similarity function does not ensure invariance to rotation and translation.

2 Q2

All the keypoint matches are not accurate. They should match as the fingerprints of the 2 test examples look very similar.But we do not expect the keypoints to systematically match correctly as they describe local features. Some may not appear in one of the 2 fingerprints due to noise or a mismatch in the captured region of the finger. This last scenario could also be caused by the segmentation or other preprocessing techniques that eroded the border of the fingerprints. Depending on the discriminatory power of the feature descriptors and on the choice of distance metric (here, the normalized Hamming distance), similar local features could appear at different locations in the images.

Despite that, if we choose a good similarity metric, we can effectively filter out the effect of wrong matches and discriminate the fingerprints.

We chose to rely on the opposite of the summed distances of the 30 best local keypoint matches as described in the code below. (Note that the list *matches* is already sorted by increasing distance)

```python
def local_img_similarity(matches):
    N = 30
    N_best_matches = matches[:N]
    total_distance = 0
    for match in N_best_matches:
        total_distance += match.distance

    return -total_distance
```

It allows us to differentiate a group of 6 candidate fingerprints from the others.

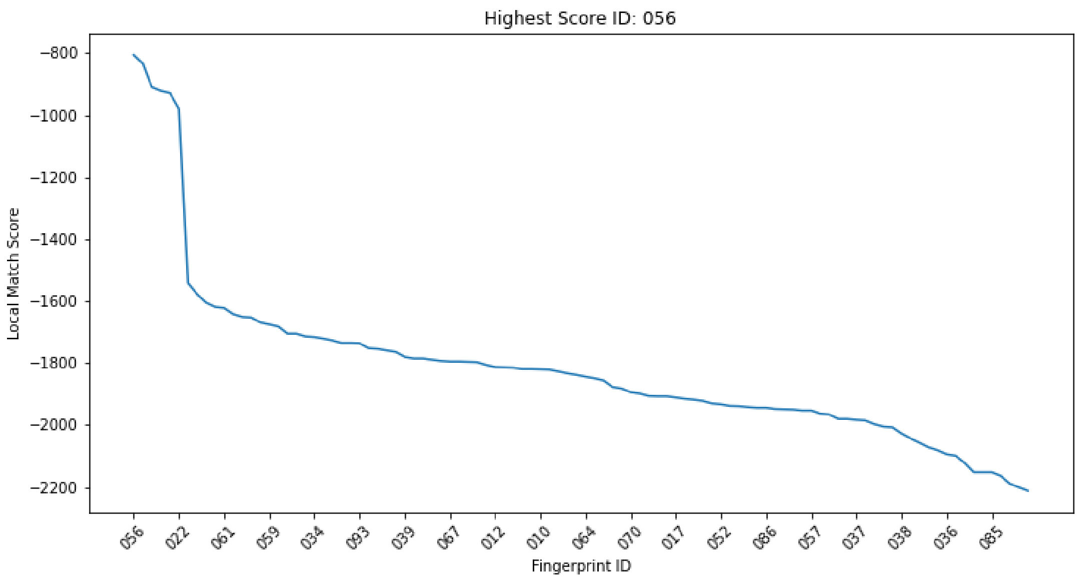


FIGURE 2 – Local Match Scores for Different Fingerprints

3 Q3

As suggested in the instructions, I opted for counting the number of matches such that the euclidean distance between the reduced sets of keypoints remains below a fixed threshold, here 5 (see the code below).

```python
from scipy.spatial import distance
def global_img_similarity(matches, reg_kp1, kp2):
    total_correct_matches = 0
    for match in matches:
        total_correct_matches += distance.euclidean(reg_kp1[match.queryIdx].pt, kp2
    [match.trainIdx].pt) < 5
    return total_correct_matches
```

4 Q4

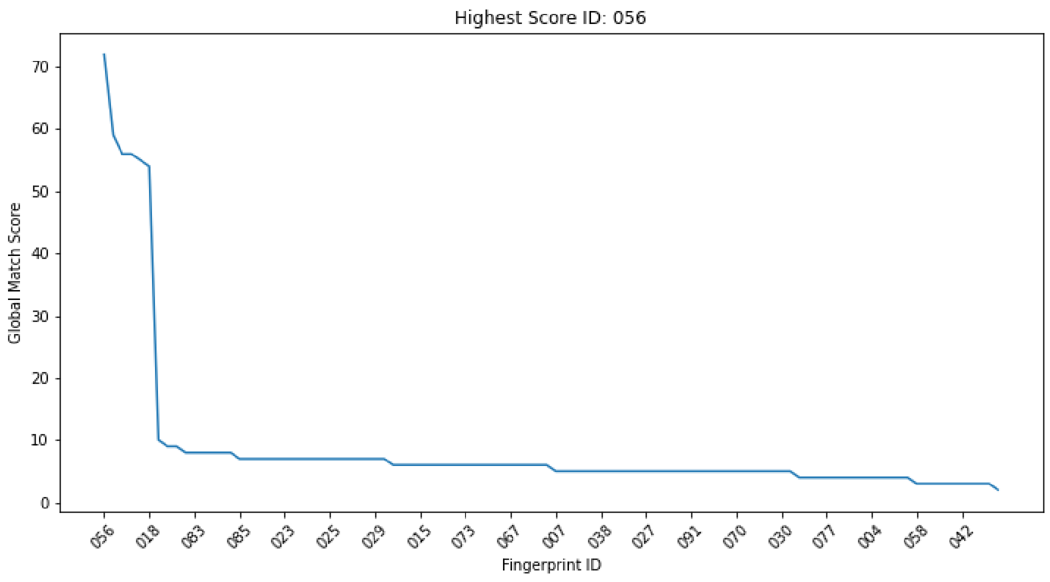


FIGURE 3 – Global Match Scores for Different Fingerprints

The threshold can be drawn after the fingerprint ranked 6th as the score significantly drops after it. Given that the 6th fingerprint has a score of 54 while the 7th has a score of 10. We could set the threshold to any value between those two, let us choose the average of the two : $\frac{54+10}{2} = 32$.

5 Q5

We separately compute the global and local similarity score and sum them together.

The local similarity is chosen to be the number of matched keypoints that have a distance below 85. The global similarity is chosen to be the number of global matched keypoints, after affine transformations (estimated with RANSAC), for which their euclidean geometric distance is lower than 5.

```
from scipy.spatial import distance
def hybrid_img_similarity(matches, kp1_reg, kp2):
    total_correct_local_matches = 0
    for match in matches:
        total_correct_local_matches += match.distance < 85

    total_correct_global_matches = 0
    for match in matches:
        total_correct_global_matches += distance.euclidean(kp1_reg[match.queryIdx].
   pt, kp2[match.trainIdx].pt) < 5

    return total_correct_local_matches + total_correct_global_matches
```

The hybrid matching score approach (see figure 4) is not far better than the global or local ones alone although the threshold 5 and 85 have been chosen for both the local and global similarity to contribute more or less evenly to the final score.

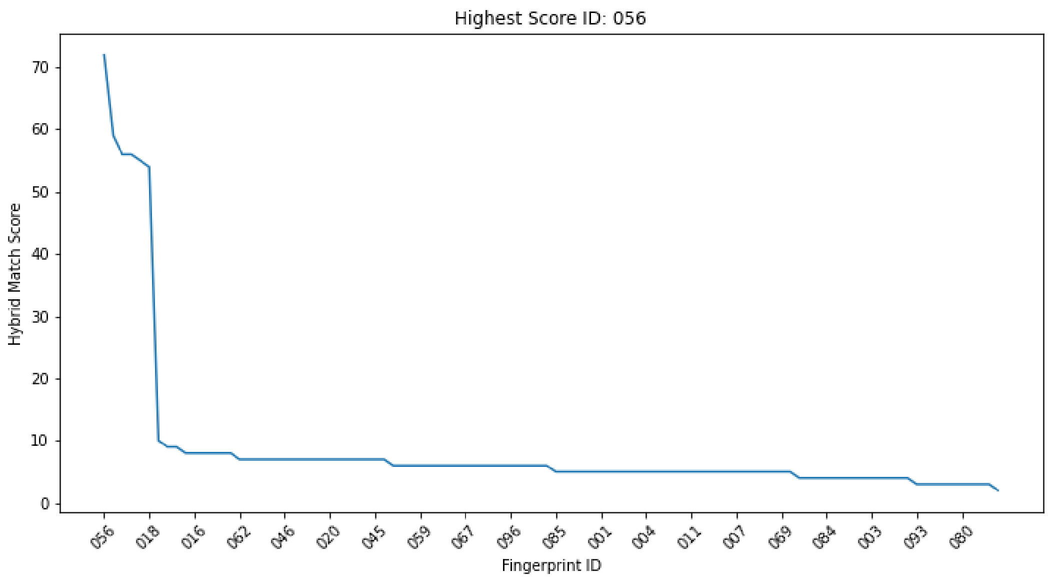


FIGURE 4 – Hybrid Match Scores for Different Fingerprints

They however allow us, as it was already the case for the local and global matcher, to clearly set the 6 first fingerprints (6th fingerprint score : 111) apart as the score significantly drops from the 7th fingerprint (score : 27) onwards. We can set a threshold between 27 and 111, say $\frac{27+111}{2} = 69$ for the identification system.

6 Q6

Looking at *iris_perpetrator.bmp*, we can see that several factors could hinder the segmentation of the iris and the matching procedure : image resolution, sight direction, reflection, eyelashes, pupil dilation, the opening of the eyelids.

In order to be robust to occlusion and possible rotation and translation, it would seem more promising to rely on local features matching instead of global. However, if we were to spend more time on iris localisation and segmentation and pattern alignment, global features would prove to be more distinctive and reliable as local structures of the iris are redundant, i.e. repeat themselves at different locations.

7 Q7

After trying different local and global procedures, I opted for local matches where the similarity score would be the opposite of the summed distances of the 10 best matches.

```
def local_img_similarity(matches):
    N = 10
    N_best_matches = matches[:N]
    total_distance = 0
    for match in N_best_matches:
        total_distance += match.distance
    return -total_distance
```

Although it is not a perfect solution in terms of discriminative power (as can be seen in figure 5), we do not expect irises to be as discriminating as fingerprints given the difficulties mentioned in Q6.

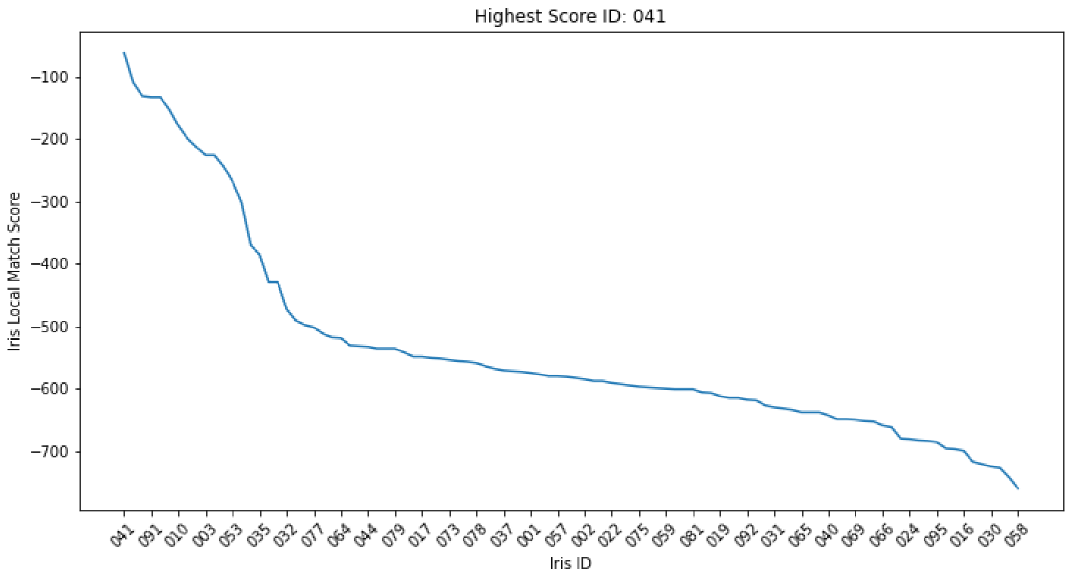


FIGURE 5 – Local Match Scores for Different Irises

We can in fact observe that there is only one common id in the top-10 ranked irises and the top-10 ranked fingerprints (from the hybrid similarity) : ID 056. It ranks first in the fingerprints but only eighth in the irises.

For all the aforementioned reasons, I would not rely on the iris similarity score alone to accuse a suspect. But ID 056 seems suspicious if we additionally take fingerprints into account.

8 Q8

To fuse both biometric systems (local similarity scores for the irises and hybrid similarity scores for the fingerprints) at the score level, I decided to normalize independently both scores in the range $[0, 1]$ and sum them together. In this manner, none of the 2 scores is played down by the other.

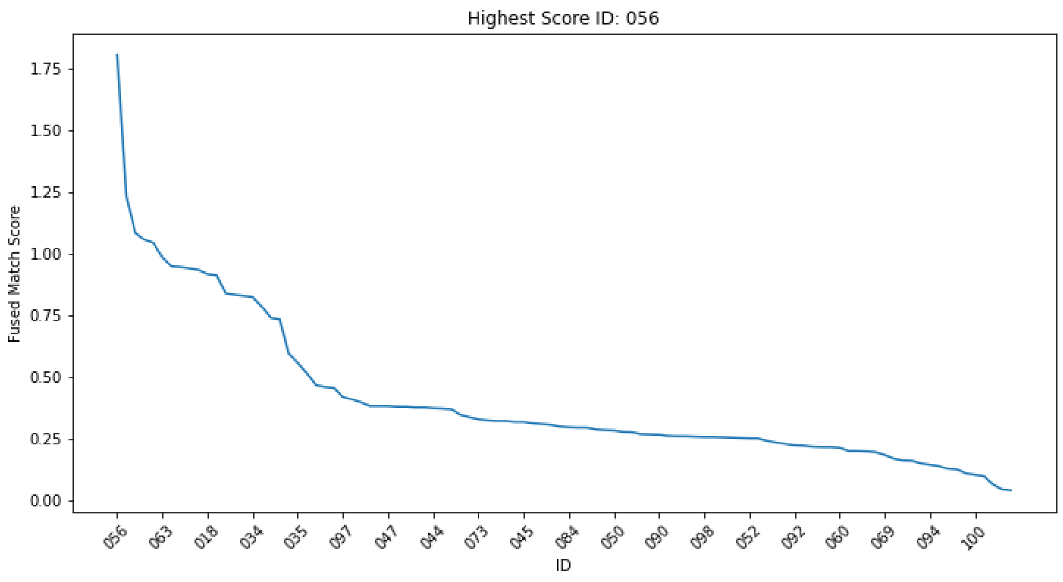


FIGURE 6 – Fused Match Scores for Different Iris-Fingerprint Pairs

If we refer to figure 6, the perpetrator would thus be ID 056.

I do feel confident in my prediction : ID 056 is the only common ID in the top-ranked fingerprints and irises (as said in Q7), the fusing procedure is fair, ID 056 scores very high (1.80) compare to the maximum possible score (2.0), the difference between the first and second top-ranked matches in the fused score is significant (1.80 v.s. 1.23) on a range of possible values of $[0, 2]$.

9 Deep Learning-Based Fingerprint Identification System

Create a machine learning based identification/authentication system and evaluate it (6pt).

9.1 Fingerprint Classification DNN Training

I decided to develop a deep-learning model trained on classifying fingerprints. The code for training the model can be found in the notebook *fingerprint-identification-system.ipynb*

The data set comes from kaggle and can be found at `https://www.kaggle.com/datasets/peace1019/fingerprint-dataset-for-fvc2000-db4-b`. It consists in 80 fingerprints per individual, 10 individuals, a total thus of 800 samples. 75% of the dataset is used for training and the 25% remaining defines the validation set. The split is done at random and in a stratified way such that the number of samples for each individual is proportionally divided among both sets.

A ResNet50V2 model pretrained on ImageNet where we remove the top layers is used as backbone. The parameters of the backbone layers will remain fixed. On top of it, we add our own layers that we will train : a GlobalAverage2D layer to reduce the number of dimensions by averaging, a dense layer and a softmax layer. The loss is chosen to be the cross-entropy loss. **??** depicts the layers of the architecture following the format *<layer or loss> (<input.shape>)*

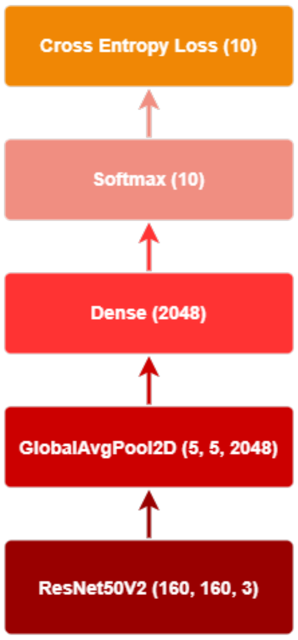
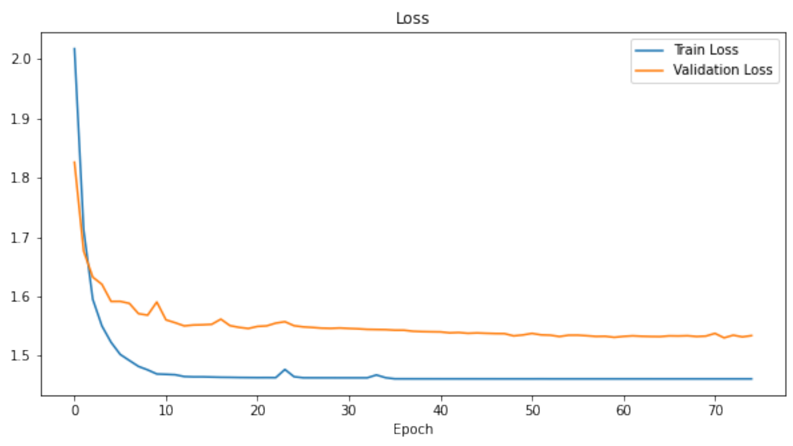


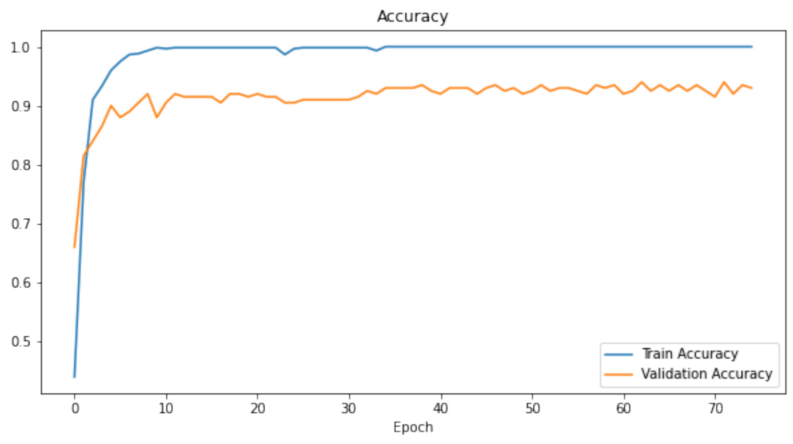
FIGURE 7 – Fingerprint DNN Classifier Architecture

We rely on Adam as optimizer with a starting learning rate of 0.001. We however apply an exponential decay to the learning rate after each epoch with a decay rate of 0.99. The batch size is uncommonly chosen to be 1. It puts more emphasis on the stochastic behaviour of SGD (or rather Adam in our case), i.e. more emphasis on exploring the parameter space. It can lead to less consistent results and thus more difficulties in tuning the training hyperparameters. It however proved to deliver the best accuracy in our experiments.

We run the training for 75 epochs and save the best model in the file *fingerprint-recognition.h5*. This model achieves an accuracy of 94% on the validation set. The training and validation loss and accuracy curves can be found in figure 8.



((a)) Cross-Entropy Loss



((b)) Accuracy

FIGURE 8 – Train and Validation Loss and Accuracy

9.2 NN-Based Feature Extraction

From the trained neural network, we remove the last layer, i.e. the softmax layer. Running this truncated model on any sample will give us a vector of numbers which we call the embedding of the current fingerprint. This embedding is supposed to be the result of a classifier that highly discriminates fingerprints of different individuals but still recognize similar fingerprints. We thus assume that the embeddings will keep this property.

9.3 NN-Based Fingerprint Identification System

We will now retrieve the fingerprint embeddings from the initial dataset of this assignment. We can apply a similarity measure to compare the embeddings and rank the most similar ones to the perpetrator's. The similarity measure is as follows

```
mss = lambda x,y: 1/(1+np.square(x-y).mean())
```

and the match scores are depicted in figure 9.

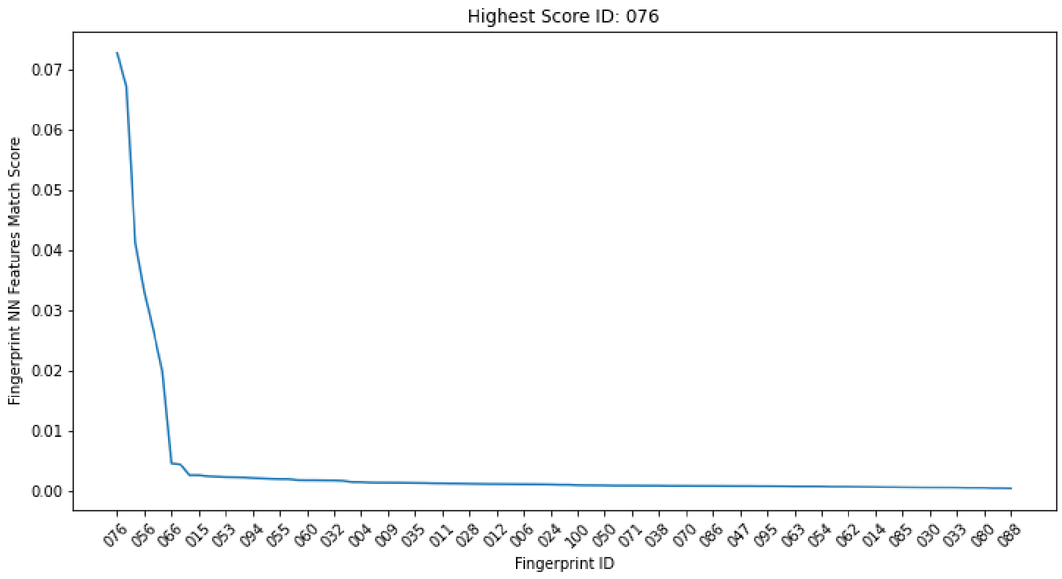


FIGURE 9 – NN-Based Feature Match Scores for Different Fingerprints

We observe a very efficient discriminatory power of the neural network embeddings that allows to single out the 6 fingerprints that best match.

We can point out that after the 6 best matches an elbow appears in the curve. We show in figure 10 the perpetrator's fingerprint and in figure 11 the 14 best matches.

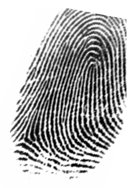


FIGURE 10 – The Perpetrator's Fingerprint

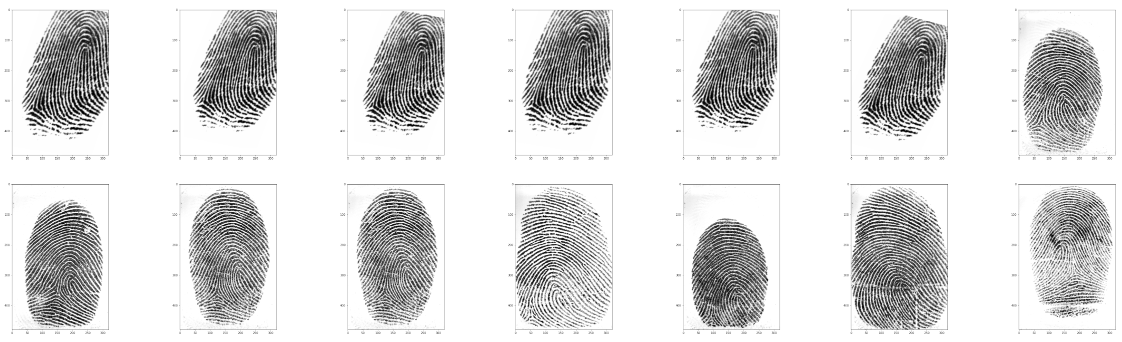


FIGURE 11 – The Best Fingerprint Matches With our DNN Approach

We see a coherent decrease in similarity as we scan them in order. We also observe that the 6 best matches are very similar to the point that a human eye would consider that they come from the same individual.

9.4 Comparison with Traditional Computer Vision

It is interesting to compare this approach to our hand-made hybrid fingerprint identification system based on ORB keypoints extraction. The results are very good and both methods have there own advantages. The machine learning method depends a lot on the quality of the training data and the training itself. It has the potential to extract more valuable fingerprints. Broadening the training dataset would allow for better generalization and adaptation. On the other hand, the traditional approach does not require any prior data or training and can directly be used to extract features. Keypoint extraction has unfortunately a limited discriminatory power which can be a bottleneck in certain applications. Furthermore, it is good to note that local features are not as robust as global features although global matching requires a thorough procedure to cope for instance with rotation and translation. Machine Learning approaches can have a hard time handling rotation as well except if the data set contains samples in different rotations. CNN provides however translation equivariance thanks to weight sharing in convolutional layers.