# Detection and Identification of Roman Emperors Using Facial Recognition

Karel van Klink
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
k.vanklink@student.utwente.nl

## ABSTRACT

Facial recognition is a much developed field of research, but the technique has not before been applied to recognizing Roman Emperors' statues. In this research we will attempt to classify and identify statues and busts of Roman Emperors using existing techniques, and discuss what features of these statues cause different results compared to pictures of human faces. The results are ROC curves for all the used test scenarios, and matrices with Euclidean distances among all the faces of Roman Emperors in the dataset that was used. From this we conclude that Roman Emperors can be distinguished using existing techniques, albeit at inferior performance compared to distinguishing human faces. We also conclude that details such as a missing tip of the nose or varying levels of detail in the eyes and pupils pose a major influence on the result of the classification by the existing solutions used.

## Keywords

Face detection, Facial recognition, Neural network, DLIB, Facenet

## 1. INTRODUCTION

For hundreds of years, the majority of Europe was part of the Roman Empire. Ruled by numerous Emperors, who were depicted in statues and busts all over the different provinces. During these times, Roman sculptors may very well have used meticulous directions for making these sculptures. With new techniques available in the fields of texture classification and facial recognition, we now have the means to (dis) confirm this theory. This research will seek to answer the questions regarding the working methods of Roman sculptors, how precise their working methods were, and how similar the busts and statues of different Roman Emperors really are, from a facial recognition perspective.

This research will continue where research on facial recognition for paintings and police sketches left off [11, 9, 18]. The situations presented in these papers are similar, in the sense that they all use pictures that aren't a picture of a person, but an 'artist impression'. This is also the case for the sculptures of Roman Emperors. What makes this research different however, is the fact that the sculptures do not have the same texture that the sketches and paintings have.


*31$^{st}$ Twente Student Conference on IT* July 5$^{th}$, 2019, Enschede, The Netherlands.


## 2. RESEARCH QUESTIONS

This paper answers the following research questions.

**RQ1** Can existing facial recognition solutions be used to recognize and identify sculptures of Roman Emperors?

- **RQ1.1** What is the performance of existing solutions for identifying Roman Emperors, compared to identifying human faces.
- **RQ1.2** What differences between statues of Roman Emperors and pictures of human faces significantly influence the performance of the classifiers?

## 3. RELATED WORK

Much research has been done on texture analysis, facial recognition, and facial expression recognition [8, 19, 10, 5]. Facial recognition in the case of *cooperative user scenarios* [7] has been readily applied in real-life situations such as - but not limited to - facial recognition systems at airports for automated border control gates [12]. It has to some extent been applied to statues in work done by Tyler et al. using a 3D morph-able model [17], to police sketches [18, 9], and to paintings made throughout history by Rosa et al. [11]. However, the work done by Rosa et al. used a proprietary solution by now Facebook-owned Face.com [15, 16] and can therefore not be replicated.

The research that has already been done was presented with a situation that is similar to this research. The statues of Roman Emperors are not true pictures taken of the actual Emperors themselves, but are artistic impressions. The difference is in the way this has been done. Both the paintings and sketches in the research done by Tyler et al. [17] and Rosa et al. [11] respectively, used depictions of people that have a texture that is more true to nature. For the Roman Emperors this is not the case. This research will contribute by applying facial recognition to the sculptures of Roman Emperors, in order to investigate whether existing classifiers can be applied to these statues, and how these solutions can be improved upon in order to increase the accuracy of the classifiers when used on these statues.

## 4. BACKGROUND

For the last few decades, numerous techniques have been proposed to perform facial recognition [1]. These techniques can be divided into two categories: trained and

untrained. In this paper, we will focus on using trained techniques. We will make use of two different classifiers: `DLIB` [6] and `Facenet` [13].

## 4.1 datasets

### *Roman Emperors*

The dataset of Roman Emperors used in this research consists of 140 pictures, almost all of which are taken from a frontal perspective. The dataset contains some pictures under different lighting conditions, and with the subject not facing the camera directly. These are assumed to be only minor problems. What will be a bigger challenge however, is dealing with other situations that would not occur in a more common recognition scenario. Things like a lack of pupils or cracks covering the face are among potential issues for a classifier to accommodate for.

Some of the potentially problematic situations are shown in the pictures in Figure 1. From left to right we have the following pictures with their corresponding caveats. The first picture shows a (relatively) ideal situation: no cracks on the sculpture, clear lighting and no missing chunks as a result of damage done to the bust. In the second example we see a situation that should not pose a problem to most modern classifiers: low illumination. The third picture shows a face with a very rough texture, as opposed to the more smooth faces in the other pictures. This could result in mixed results depending on the classifier used, since it will have a different effect on LBP or LDP based classifiers. These kinds of classifiers make use of local changes in texture, thus possibly resulting in a different outcome depending on the contrast present in a picture.

The dataset also contains statues that contain other blemishes, such as a missing or damaged nose, damaged eyes, or even missing a substantial part of the *inner face*. Given the fact that these consist for only roughly one third of the total dataset, we choose to disregard these pictures. By doing this, we will have higher performance when using existing classifiers, and will yield better results when improving upon these.

**Figure 1. Samples from the used dataset that display different encountered situations when classifying**

### *Human Faces*

To compare the results of the recognition of the Roman Emperors, we will use a dataset consisting of frontal pictures of human faces. The dataset we use is part of the FRGCv2 dataset [2]. Specifically, the Spring-2003 subset consisting of 4110 pictures of 150 different people.

Because the lighting conditions are similar for both datasets, and the fact that all pictures in both datasets are taken from the front, we will be more likely to make a satisfactory comparison between the results of the datasets.

## 4.2 Evaluating the results

For evaluation of the results of the classifiers' performances, we will make use of Receiver Operator Curves (ROC) and their corresponding Area Under Curve (AUC) values. "[An ROC] is a graphical representation for displaying the transition between $TPR$ [True Positive Rate] and $FPR$ [False Positive Rate]. $TPR$ indicates correctly classified or total positive values and [are] plotted on the y-axis, whereas $FPR$ indicates incorrectly classified or total negative values plotted on the x-axis" [14]. Classifiers with higher performance will have points close to $TPR = 1$ and $FPR = 0$, thus the area under that ROC (the AUC value) is close to 1. "The area under the ROC curve measures the probability [...] of the two images [...] to be correctly identified" [4].

# 5. METHOD

## Existing Solutions (RQ1.1)

In order to verify and compare the performance of existing solutions on recognizing Roman Emperors, we will test on both the datasets of the Roman Emperors and the FRCGv2 subset. On both these datasets, we use both `DLIB` [6] through a wrapper library `face_recognition` [3], and Facenet [13].

### *Measuring performance*

To measure the performance, we first prepare the dataset of the Roman Emperors. We make a selection by hand on the pictures that are not damaged, and remove the pictures of Emperors that are missing a nose, or other parts of their face. This will leave us with 82 pictures of 26 emperors. After this, we remove duplicate pictures from the remainder of the dataset, resulting in 79 pictures of 26 emperors. We remove the duplicated, because these would have resulted in unrealistic results. Any classifier would be able to recognize two identical pictures. We make the selection on intact statues to be able to properly compare them using existing solutions, because the models for these networks have been trained on faces with all facial features present. The other dataset is kept as it is, and contains 4110 pictures of 150 different people.

After making the selection, we have two datasets that contain pictures that are not aligned. From these two sets, we create two new datasets that contain pictures that are all aligned. This is done using the `dlib.frontal_face_detector` and `imutils.face_utils.FaceAligner`. In the dataset of the Roman Emperors, one picture had to be manually aligned because it was not recognized by the `frontal_face_detector` which is `keizer (62).jpg` as shown in Figure 2. The alignment is done using the algorithm found in Algorithm 1.

With these four datasets, we now use the two recognizers to test their performances. From both `DLIB` and `Facenet` we get the facial embedding data stored in a vector in Euclidean space, and calculate the distances between them. This will result in eight confusion matrices of all the distances between all the faces. From these matrices we compute eight ROC curves, with their corresponding AUC-values that will display the performance of the recognizer on each set of faces. With these results, we can satisfyingly answer research question 1.1.

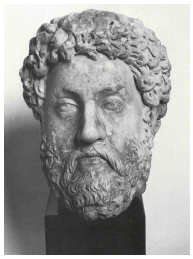


**Figure 2. A statue of Emperor Commodus that could not be automatically aligned**

The facial features are calculated with `DLIB` by using the algorithm shown in Algorithm 2 and Algorithm 3 shows the calculations done with `Facenet`. The computation of the ROC curve with its corresponding area under the curve is calculated as shown in Algorithm 4. On top of these graphs, we will plot the TPR and FPR rates against different thresholds for the cases of the Roman Emperors and humans using `DLIB` and `Facenet` with the unaligned pictures. These graphs will allow us to determine whether a threshold exists where the TPR is close to 1, and the FPR close to 0.

## Causes of performance differences (RQ1.2)

With the computed ROC curves and AUC values, we will be able to draw a conclusion whether performance differs when using the classifiers on pictures of humans, or on statues of Roman Emperors. From this, we will perform a visual inspection on the pictures that seem to cause these discrepancies. The pictures that are to be inspected, are the ones that have been incorrectly labelled by the classifiers. These can be found by inspecting the matrix that is computed by both classifiers for all the distances between the vectors of facial features.

In this matrix, we will look for the pictures that share a relatively low distance among each other, but do not share the same labels. From these pictures, we will perform a visual inspection on the facial features that these pictures share, and compare them to the facial features of the pictures they should have matched with instead. From this, we will draw a conclusion that seeks to answer research question 1.2.

# 6. RESULTS

All the different ROC curves for the eight combinations possible between datasets, classifiers and aligned or unaligned can be found in Figure 3. Each graph is titled accordingly by the classifier used, whether the pictures in the dataset were aligned or not, and what dataset was used. Note that in the titles, the dataset of human faces is called `Spring2003_controlled`. Each graph also displays its corresponding area under the curve, or AUC value.

In Figure 4 we have the TPR and FPR values plotted against the threshold of the classification. From these graphs, we can see that for classifying human faces, both classifiers have a threshold value where the TPR is 1 and the FPR 0. For the classification of Roman Emperors, no such value exists. A similar graph can be found in Figure 5. Note that in this case, there is a range where the selection of a threshold *does* allow for classification at TPR nearly 1 and FPR nearly 0 at around 0.45 for `DLIB` and at around 0.9 for `Facenet`.

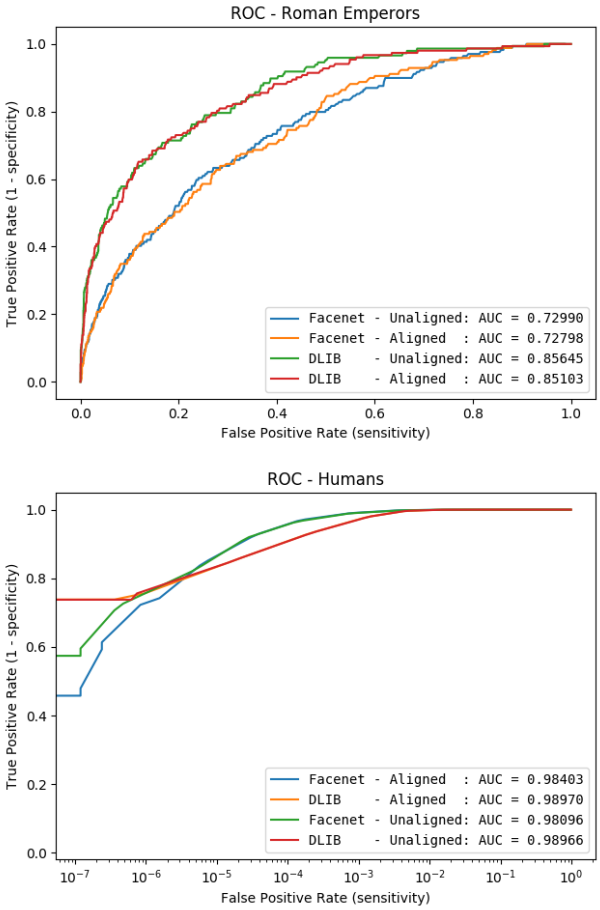



**Figure 3. Various ROC curves using different methods and datasets**

In the results, we also get the matrices of the distances between all the embeddings in Euclidean space. A part of one of these matrices is shown in Figure 6. In these matrices, we can spot some pictures with the same labels being incorrectly classified, because they have a relatively large distance between them. These pictures in particular are picture 58 considered a close match to pictures 1, 38, 64 and 90. Also, we have 127 relatively close to 52, and 36 close to exclusively 37. All despite the fact that these pictures are of different Emperors. The difference between pictures 37 and 138 compared to the other pictures of the same Emperor can be found in Figure 6. The complete matrix, and all the other matrices from the different test scenarios can be found in Appendix A.

## 6.1 Discussion

### *Performance of classifiers*

From the results in Figure 3 it becomes clear that the performance of the classifiers heavily depends on the dataset that is used. Both `DLIB` and `Facenet` have far superior performance when working with the dataset of human faces, compared to the Roman Emperors. On top of that, there is the performance difference between the two classifiers. The AUC values of 0.856 and 0.851 of `DLIB` compared to the same values in `Facenet` of 0.730 and 0.728 vary significantly with relative differences of 11% and 10% for unaligned and aligned pictures respectively. However, because of the fact that both classifiers also perform align-

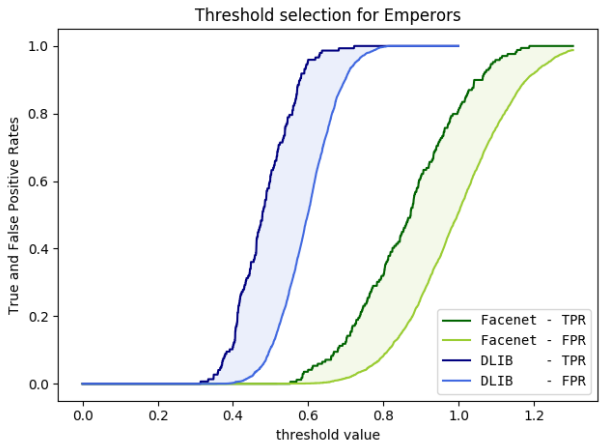



**Figure 4. TPR and FPR at various thresholds when classifying Roman Emperors**

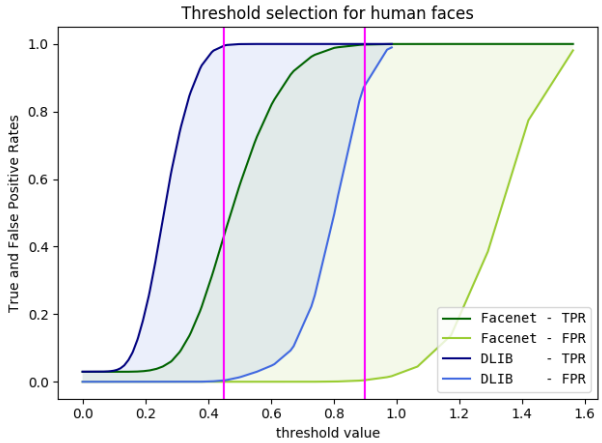



**Figure 5. TPR and FPR at various thresholds when classifying humans**

ment on the input pictures internally when calculating the embeddings, the fact that the differences between unaligned and aligned pictures are negligible is to be expected.

### *Causes of incorrect classification*

Looking at the results in the matrix, we will perform a visual inspection on the incorrectly labelled pictures that share a relatively low mutual distance between their embeddings.

First, we have picture 58 ranking closest to pictures 1, 38, 64 and 90. Upon inspection of these pictures, the reason for this to happen becomes obvious. In all these pictures the subjects are looking in the same direction, and both Emperors share a similar style of facial hair. Pictures 58 and 90 are shown side-by-side in Figure 7. However, picture 58 is the only picture that contains Emperor Clodius Albinus. (This situation also arises with picture 131 for example, since it is the only picture of Emperor Vespasian) Therefore, the closest match never could have been a correct one. Despite this fact, the mutual distances are of such proximity that they fall within the same range as for actual matches. If classified correctly, the pictures of 'single' emperors should all have relatively large distances from the other faces in the set. This is not the case for

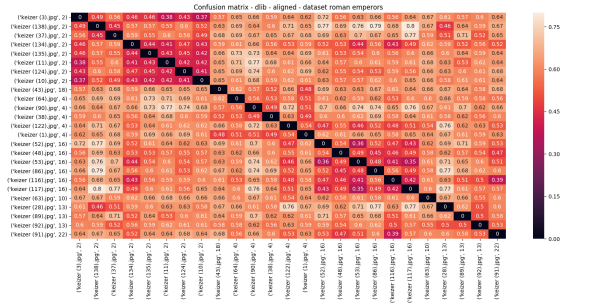



**Figure 6. Part of the matrix of Euclidean distances between faces detected in the dataset of Roman Emperors**

most of the Emperors that only have a single picture in the dataset.

**Figure 7. Statues of Emperors Clodius Albinus and Caracalla. The statue of Clodius Albinus is incorrectly classified as a depiction of Caracalla**

Next, there is the case of pictures 127 and 52 of Emperors Tiberius and Marcus Aurelius. Picture 127 was incorrectly classified as a depiction of Emperor Marcus Aurelius, despite the fact that 4 more pictures of Tiberius were in the dataset. Their distances to picture 127 were greater than the distance to picture 52. Upon inspection, we notice that picture 127 and 52 do indeed share some facial features to a certain extent. Both are shown in Figure 8. Here we can see that both faces share a similar shape, and that in both cases the pupils are missing. The statues might not contain any retinas or contrasted pupils, there are cases where the retinas are outlined which might help in classification. Here, this is not the case.

Also, there is the case of pictures 37 and 138 that poses an interesting result. These pictures are both of Emperor Augustus, and they are labelled correctly amongst the two of them. However, they are both relatively distant from the other pictures of Emperor Augustus. Visual inspection of these two pictures, compared to the other pictures of Emperor Augustus, immediately shows the root cause of this incorrect classification: both pictures 37 and 138 are missing the tip of their noses, and they are the only pictures of Emperor Augustus in the dataset where this is the case. Because of this, the classifiers will try and map the mark for the tip of the nose, but it will be placed higher than it should have, if these statues were to still have their noses fully intact. This will result in different embeddings for these pictures, and from this it becomes

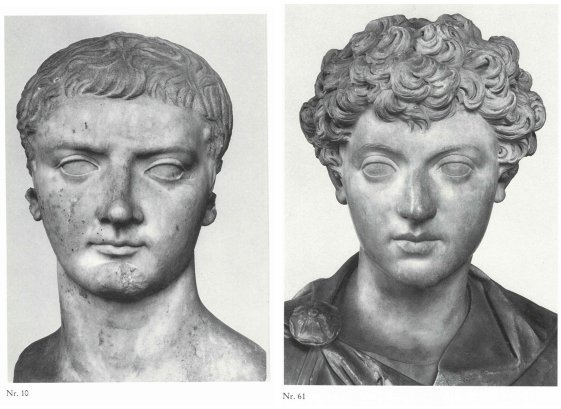


**Figure 8. Statues of Emperors Tiberius and Marcus Aurelius, where this statue of Tiberius is incorrectly classified as Marcus Aurelius**

apparent why they are classified almost separately from the other statues of Emperor Augustus. Both pictures 37 and 138 are shown in Figure 9.

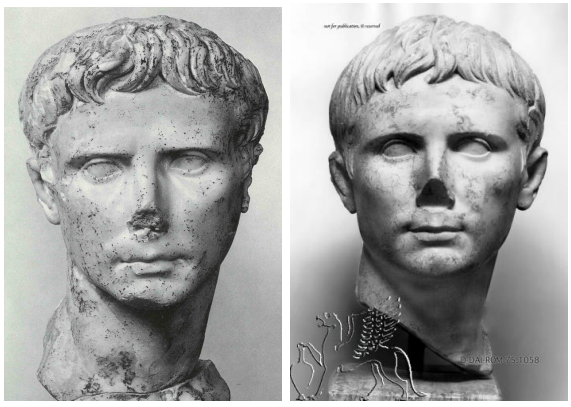


**Figure 9. Statues of Augustus that are classified separately from the other statues of Augustus**

# 7. CONCLUSIONS

With the results found in the ROC curves and their corresponding AUC values, we can conclude that existing classifiers are sufficiently able to distinguish statues of Roman Emperors. However, these results still are significantly lacking when compared to the results on human faces. This is most likely due to the fact that these classifiers have never been trained on pictures of Roman Emperors. A way to improve these results would be to have the networks train on pictures of Roman Emperors instead of human faces.

Because of the limited size of the dataset of Roman Emperors, there were many cases of Emperors incorrectly classified simply because of the fact that they appeared as single pictures in the set. An improvement for further work would be to construct a dataset that contains more pictures of Roman Emperors that do not contain many blemishes. As could be seen for the pictures in Figure 9, a common imperfection can cause a separation in classification. Also, classifications appear to be heavily influenced by details such as the pupils and overall details of the eyes. For human faces, this is a very valid feature to train a network on, but for the Emperors' statues, these can vary significantly per Emperor. This could be circumvented by training a network on pictures of Emperors in such a way where details in the eyes are found to be much less important, or by pre-processing the pictures in such a way that the eyes are the same for all the statues.

# 8. REFERENCES


[1] K. Chengeta and S. Viriri. A survey on facial recognition based on local directional and local binary patterns. In *2018 Conference on Information Communications Technology and Society (ICTAS)*, pages 1–6, Durban, Mar. 2018. IEEE.

[2] P. A. Flanagan. Face Recognition Grand Challenge (FRGC), Dec. 2010.

[3] A. Geitgey. The world's simplest facial recognition api for Python and the command line: ageitgey/face_recognition, May 2019. original-date: 2017-03-03T21:52:39Z.

[4] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, Apr. 1982.

[5] T. Jabid, M. H. Kabir, and O. Chae. Local Directional Pattern (LDP) for face recognition. In *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*, pages 329–330, Jan. 2010.

[6] D. E. King. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[7] S. Z. Li and A. K. Jain, editors. *Handbook of face recognition*. Springer, London : New York, 2nd ed edition, 2011.

[8] W. Liu, S. Li, and Y. Wang. Automatic Facial Expression Recognition Based on Local Binary Patterns of Local Areas. In *2009 WASE International Conference on Information Engineering*, volume 1, pages 197–200, July 2009.

[9] S. Pramanik and D. Bhattacharjee. Geometric feature based face-sketch recognition. In *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)*, pages 409–415, Mar. 2012.

[10] A. R. Rivera, J. R. Castillo, and O. O. Chae. Local Directional Number Pattern for Face Analysis: Face and Expression Recognition. *IEEE Transactions on Image Processing*, 22(5):1740–1752, May 2013.

[11] J. d. l. Rosa and J.-L. SuÃąrez. A Quantitative Approach to Beauty. Perceived Attractiveness of Human Faces in World Painting. *International Journal for Digital Art History*, (1), June 2015.

[12] J. Sanchez del Rio, D. Moctezuma, C. Conde, I. Martin de Diego, and E. Cabello. Automated border control e-gates and facial recognition systems. *Computers & Security*, 62:49–72, Sept. 2016.

[13] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015. arXiv: 1503.03832.

[14] M. Sundaram and A. Mani. Face Recognition: Demystification of Multifarious Aspect in Evaluation Metrics. *Face Recognition - Semisupervised Classification, Subspace Projection and Evaluation Methods*, July 2016.

[15] Y. Taigman and L. Wolf. Leveraging Billions of Faces to Overcome Performance Barriers in

Unconstrained Face Recognition. *arXiv:1108.1122 [cs]*, Aug. 2011. arXiv: 1108.1122.

[16] A. Tsotsis. Facebook Scoops Up Face.com For $55-60m To Bolster Its Facial Recognition Tech (Updated).

[17] C. W. Tyler, W. A. P. Smith, and D. G. Stork. In search of Leonardo: computer-based facial image analysis of Renaissance artworks for identifying Leonardo as subject. In *Human Vision and Electronic Imaging XVII*, volume 8291, page 82911D. International Society for Optics and Photonics, Feb. 2012.

[18] R. G. Uhl, N. d. V. Lobo, and Y. H. Kwon. Recognizing a facial image from a police sketch. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 129–137, Dec. 1994.

[19] H. Zhou, R. Wang, and C. Wang. A novel extended local-binary-pattern operator for texture analysis. *Information Sciences*, 178(22):4314–4325, Nov. 2008.

# APPENDIX

## A. ALGORITHMS USED FOR PERFORMING THE EXPERIMENTS AND RESULTING MATRICES FOR ALL TESTING SCENARIOS

**Algorithm 1:** Aligning faces found in provided pictures

**Input:** *files*: list of filenames to align
**Output:** Images of aligned faces
*detector* = `dlib.get_frontal_face_detector()`
*predictor* = `dlib.shape_predictor('shape_predictor_68_landmarks.dat')`
*fa* = `FaceAligner(`*predictor*`, desiredWidth=1024)`
**foreach** *img* **in** *files*:
    *image* = `imutils.resize(cv2.imread(`*img*`), width=2000)`
    `# Convert image to grayscale`
    *gray* = `cv2.cvtColor(`*image*`, cv2.COLOR_BGR2GRAY)`
    `# Get coordinates of rectangle around the face in the picture`
    *rects* = `detector(`*gray*`)`
    `# Align the found face using the detected coordinates`
    *aligned_face* = `fa.align(`*image*`,` *gray*`,` *rects*[0])
    `# Store the aligned face to the filesystem`
    `cv2.imwrite(`*path*`,` *aligned_face*`)`

**Algorithm 2:** Calculation of facial encodings using `DLIB`

**Input:** *files*: list of filenames with faces
**Output:** *encodings*: list of all facial encodings in the files from the input
*face_encodings* = []
**foreach** *img_path* **in** *files*:
    *image* = `face_recognition.load_image_file(`*img_path*`)`
    *face_bounding_boxes* = `face_recognition.face_locations(`*image*`)`

    `# Get the encoding for the first (and only) face in the current image`
    *encoding* = `face_recognition.face_encodings(`*image*`)[0]`

    `# Append the encoding of the current face to the result list`
    *face_encodings*`.append(`*encoding*`)`

**Algorithm 3:** Calculation of facial encodings using `Facenet`

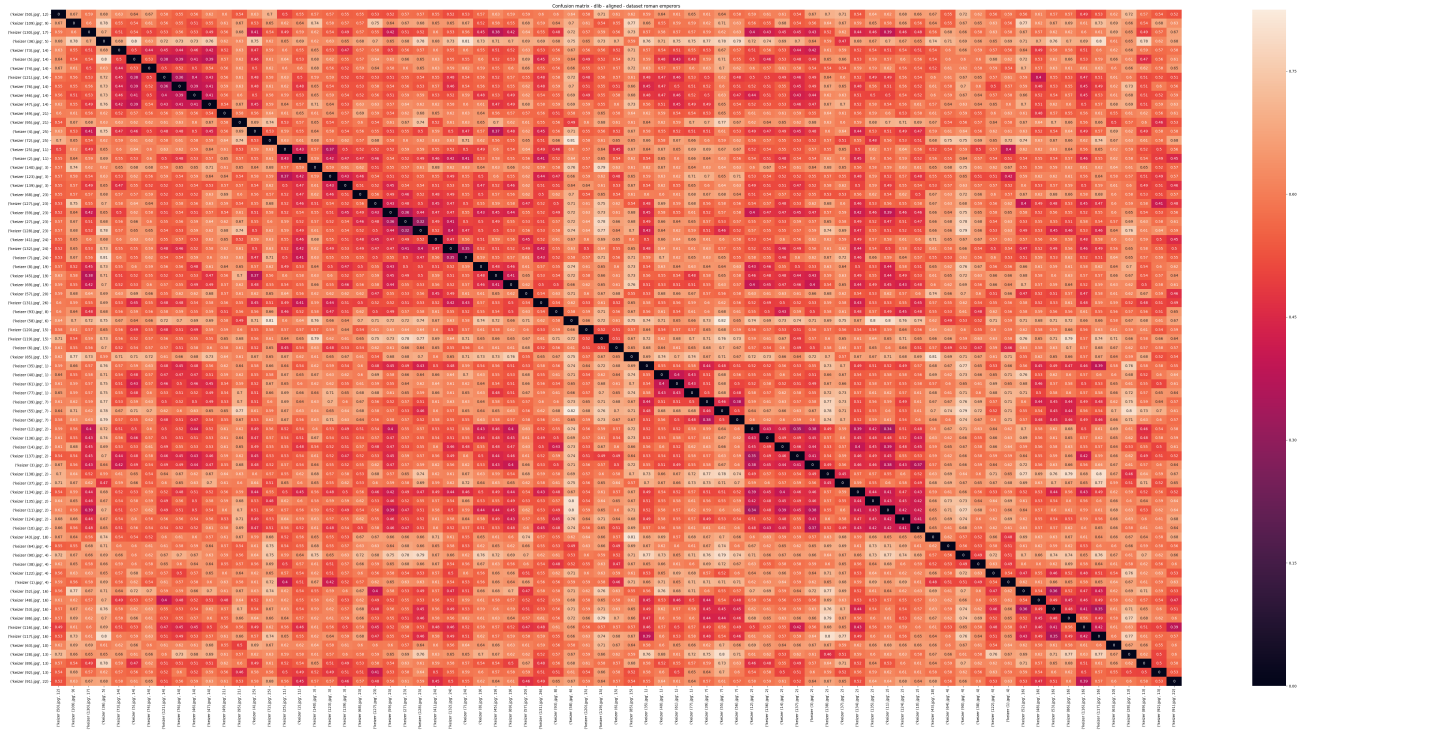
**Input:** *files*, *labels*: list of filenames with faces and their corresponding labels
**Output:** *encodings*: list of all facial encodings in the input files
*embeddings* = `tf.get_default_graph().get_tensor_by_name("embeddings:0")`
*encodings* = []
*sess* = `tf.Session()`
**foreach** *file* **in** *files*:
    *img* = `facenet.load_data(`*file*`)`
    *feed_dict* = `images_placeholder:` *img*

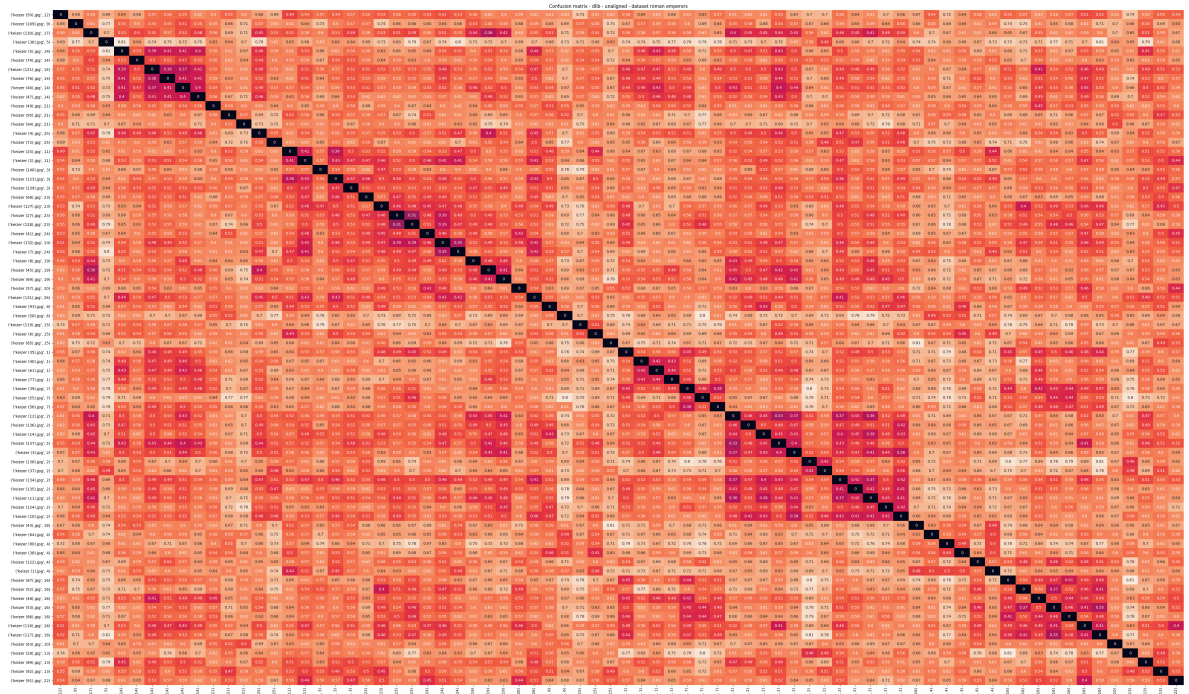
    `# Get embedding for current image`
    *embed* = *sess*`.run(`*embeddings*`, feed_dict=`*feed*$_d$*ict*`)`

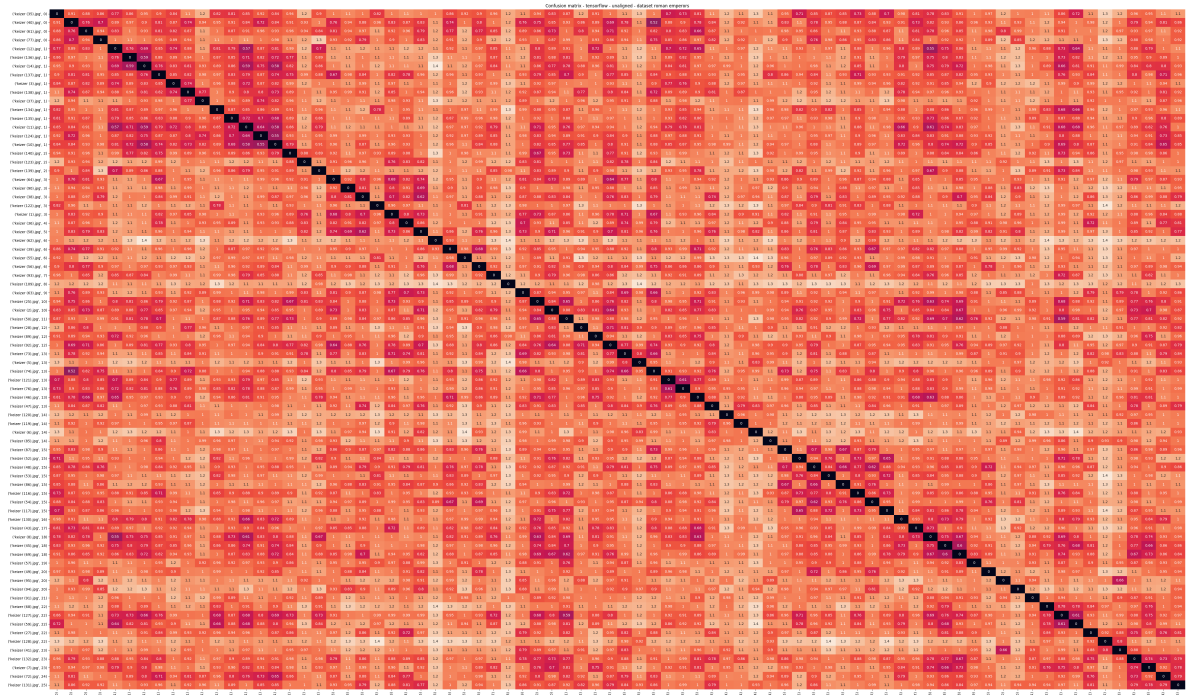
    `# Append current encoding to the result list`
    *encodings*`.append(`*embed*`)`

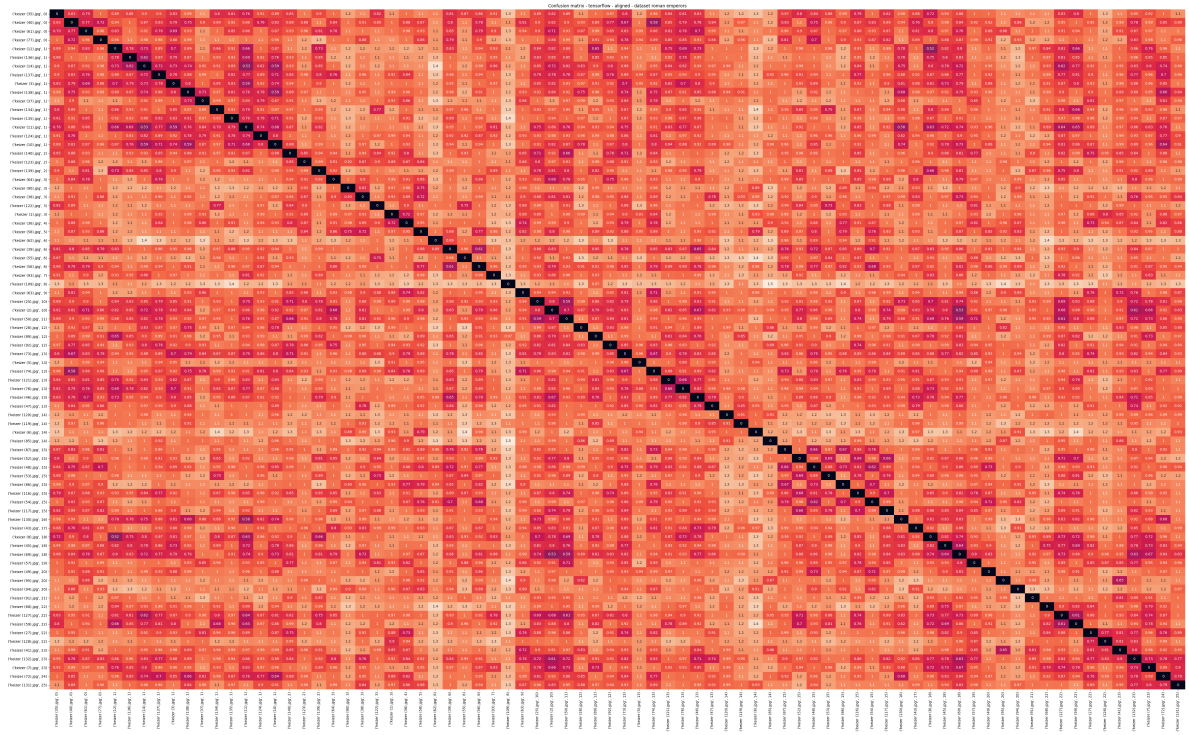
**Algorithm 4:** Computation of the ROC graph with the area under curve metric

**Input:** *encodings*, *labels*: list of facial encodings as vectors and their corresponding labels
**Output:** *points*, *auc*: list of points of the ROC curve and the auc value
*distances* = `sklean.metrics.pairwise.euclidean_distances(`*encodings*`)`
*points* = []
`# `*p*` is set to the amount of positives in the set`
`# `*n*` is set to the amount of negatives in the set`
*max_threshold* = `max(list(zip(*`*distances*`))[0])`
*threshold* = 0
**while** *threshold* <= *max_threshold*:
    *tp* = 0 `/*  amount of true positives */`
    *fp* = 0 `/*  amount of false positives */`
    **foreach** *i*, *person* **in** *enumerate(distances)*:
        `# Iterate over all the persons`
        **foreach** *j*, *match* **in** *enumerate(person)*:
            `# Iterate over every distance relative to that person`
            **if** *i* == *j*: `/* Ignore the diagonal where all distances are 0 */`
                `continue`
            **elif** *match* < *threshold*:
                **if** *labels[i]* == *labels[j]*: `/* Check if a match is true or not */`
                    *tp* `+= 1`
                **else:**
                    *fp* `+= 1`
            *tpr* = *tp* / *p*
            *fpr* = *fp* / *n*
            `# Add the TPR and FPR to the ROC curve`
            *points*`.append([`*fpr*`, `*tpr*`])`
            *threshold* `+= (`*max_threshold* `/ 1000)`
    *auc* = `sklearn.metrics.auc([`*i*`[0] for `*i*` in `*points*`], [`*i*`[1] for `*i*` in `*points*`]))`




Confusion matrix - dlib - unaligned - dataset roman emperors


Confusion matrix - tensorflow - unaligned - dataset roman emperors


Confusion matrix - tensorflow - aligned - dataset roman emperors