EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Enhancing Face Detection and Clustering for Image Analysis through an Interactive Visualization: A Broader Scope

Bachelor End Project Report

Thijmen Broeren - 1839527

supervisors:
Huub van de Wetering

6-6-2025

Abstract

Face detection and clustering have always been challenging tasks since their practical use cases can differ by a lot. The problem addressed consists of assessing how more reliable face detection models and clustering algorithms can be created, and how an interactive visualization can provide more insight into the model's correctness. Building on prior work, face detection and clustering models are improved and evaluated. For face clustering, various models of the YOLO architecture are used, a self-trained model is created using the newest YOLO architecture, and an InsightFace model is used. Of which the self-trained model performs best. For face clustering, the DBSCAN algorithm and Chinese Whispers algorithm are tested with different parameters and seven different face embedding models are tested of which the combination of the DBSCAN algorithm and the face_recognition model performed best. These models are then used in an interactive visualization allowing the user to play with these models and algorithms, and to analyze their own image dataset. The code for all experiments and the interactive visualization can be found at the GitHub[1].

[1] `https://github.com/stiverthijmen163/BEP/tree/main`

Contents

1 Introduction

Understanding how people interpret political news is influenced by multiple factors, including the way the news article is presented. When reading through political news articles, it can be interpreted differently among people. This is because of textual and visual modalities within the news. By analyzing textual and visual modalities, we can for example research whether there are certain politicians who are overrepresented compared to others within the news pages (Ruyters, 2025). Ruyters has proposed a method for this by using face detection models to detect and isolate the faces from visual modalities, and by clustering the faces (one cluster/class per unique face) as a form of face recognition. Not only can any overrepresented classes be found this way, but also any classes commonly co-occurring in visual modalities. Using the proposed solution, one could achieve more insights into the impact of textual and visual modalities on the interpretation of news content. In this thesis, the focus is on the visual modalities within news pages to find commonly co-occurring persons among them.

To understand the impact of visual modalities on the interpretation of news content it is, among others, important to have an insight as to whom are represented in representative images at the start of a news article (picture lead). While these detection models already exist to some extent, they tend to be unreliable and not robust. Therefore, this study aims to provide more reliable models and more insight into how these models work and what their strengths and limitations are.

Face detection and recognition have become more and more relevant over the years. But not only in the context of politicians in the news. Therefore a study such as this can apply to multiple applications, which is why the end product, an interactive visualization, should be easily applicable to multiple applications. For instance, to find co-occurring persons in sports images, but the scope is also broadened to future-proof the end product. Think of being able to add data as a user with additional information such as topics or most common words in a news article.

Obviously, when a model is unreliable, it can lead to wrong conclusions. This can have severe consequences if these conclusions are used for the representativeness of politicians in news picture leads since this tends to prove that certain news pages may be biased toward certain politicians. To prevent this from happening robustness and reliability are two important requirements for algorithms. Apart from creating better models, a way of improving this is by providing insight to the user as to how well a certain detection/clustering algorithm performs and allowing the user to test them in different circumstances such as noisy data or images taken at night. This can be done using an interactive visualization like the end product of this thesis.

This study focuses on improving face detection and clustering models compared to the methods introduced by Ruyters (2025) by creating different models and using more recent techniques. In addition, an interactive visualization is created to provide more insight into the model's performance and therefore the reliability of the model. Moreover, extra features will be implemented in this visualization to provide the user with new insights regarding politicians in news picture leads or any other context regarding faces in images.

An extra, but important feature, is that this study attempts to create a visualization such that it applies to different fields of study.

2 Related Work

Object Detection: According to Ali and Zhang (2024), one way of doing object detection is by using the You Only Look Once (YOLO) architecture. There are many versions of YOLO you could use for this all performing with mAP > 0.90. mAP is the mean of the average precision across all classes in a model, for more information see Solawetz (2020). Also, in Ruyters (2025) two different methods are used which do not work very well apart from each other but have increased accuracy when combined, namely haar_cascade and face_recognition. These will be used as a base model. Also, the technique of combining different models to get more detections in the database is used here. Another technique used in this thesis is InsightFace which is known for its models in face detection and recognition with high accuracy (Guo et al., 2021).

Face Clustering/Classification: When working with unlabelled data it is important to cluster the objects found by the detection models. To do this one could use K-nn and SVM as a pre-trained model and run it on the unlabelled data, but this leads to some misclassifications (Ruyters, 2025). Moreover, when clustering unlabelled data and there does not exist such a recognition model yet, K-nn or SVM would be no option. To solve this problem, there are two important factors: the embeddings used for clustering and the face clustering algorithm (Petukhova et al., 2025). For clustering, the Chinese Whispers algorithm has many applications (Biemann, 2006; Shiells and Pham, 2010; Al Ali et al., 2024). However, according to Singh and Hang (2021) and Lin et al. (2021), it can also be used for face classification and it is considered a relatively quick algorithm. One problem that arises though is that this algorithm performs better with larger datasets. The scikit-learn library has also many easy-to-implement clustering algorithms, one of which is the DBSCAN algorithm (Pedregosa et al., 2011). DBSCAN is a density-based clustering algorithm that works well for all kinds of shapes as long as the data points are close to each other. There are many ways to embed faces, one of which is FaceNet using face similarity to embed faces (Schroff et al., 2015). DeepFace introduces a way to easily implement FaceNet and some other face embedding models along with it such as VGG-Face and OpenFace (Serengil and Ozpinar, 2021). One problem that arises though is that DeepFace does not allow for inputting the locations of a face, therefore the effectiveness of a good face detection model may be reduced. One model that does allow this, is the face_recognition model (Geitgey, 2020).

Interactive Visualization: To increase the understanding of algorithms used during this study, visualizations can be used since they allow people to engage more actively with the model (Grissom et al., 2003; Naps, 2005). According to Saraiya et al. (2004) and Munzner (2014), there are specific features that are most effective for such visualizations.

Depending on the context, this means for example that a bar chart may be a better visualization instead of a pie chart and that certain colors like green and yellow should be avoided within the same line chart.

3 Datasets

To train, validate and evaluate models and algorithms worked with in this thesis, various datasets are needed. They are explained below.

3.1 Face Detection

For training and validating face detection models, a subset of two datasets is used: the Face Detection Dataset[2] and the WIDER FACE dataset[3]. The Face Detection Dataset consists of 13389 train and 3347 validation images. The images are high-quality images with one or more faces in each of them and in a large variety of scenes. The WIDER FACE dataset consists also of high-quality images, typically images contain a larger amount of faces with a greater distance from the camera divided into 61 different types of scenes. It has a total of 12880 train and 3226 validation images. Both datasets come with the bounding boxes for all faces in their respective images. Some example images are shown in Figure 1. Using both these datasets the following two datasets are created:

FDD-WF small subset: This dataset is used for training face detection models in a quick matter. It consists of 500 random training images from both datasets and 100 random validation images from both datasets.

FDD-WF subset: This dataset is a larger version of the 'FDD-WF small subset' dataset such that it can be used for training a final version of the face detection model. It contains 2500 random training images from both earlier mentioned datasets and 500 random validation images from both those datasets.

[2] `https://www.kaggle.com/datasets/fareselmenshawii/face-detection-dataset`
[3] `https://huggingface.co/datasets/CUHK-CSE/wider_face`

Figure 1: Some examples of the Face Detection Dataset (left) and the WIDER FACE dataset (right).

FD-test: For the test data, a subset of the images scraped from the internet by Ruyters has been used. Only the images containing politicians are used to significantly decrease the size of the test data, this is done because these images must be labelled by hand. Therefore, this dataset contains 290 images used in news headers. Some examples are shown in Figure 2.

Figure 2: Some examples of the FD-test dataset, original images from https://www.nu.nl/vragenoproep/6289454 & https://nos.nl/artikel/2491240.

3.2 Face Clustering

CFI: The Celebrity Face Image Dataset[4] contains high-quality photos of 17 distinct celebrities. This is overall a balanced dataset, all classes except one have 100 images each, the other class has 200 images.

Embedding on the Wall: As a second dataset for face clustering, a more realistic and less ideal dataset is used. Ruyters created a dataset[5] with all faces cut out from the news images, this includes a class called 'unknown' which will not be used, since this adds too much noise to the dataset to establish a good result when clustering. This dataset is unbalanced and contains images both of low and high quality and resolution. The number of occurrences per class ranges from 1 (Edson Elf) to 66 (Pieter Omtzigt).

Figure 3: Some examples of the Celebrity Face Image Dataset (left) and the Embedding on the Wall dataset (right).

4 Methodology

The main result of this thesis consists of an interactive visualization in which users will be able to understand how well the models and algorithms perform and are able to influence the results of such models, as explained in section 4.3. To come to this visualization, a few steps are needed to create such models and algorithms, this is visualized in Figure 4.

[4] https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset

[5] https://github.com/wieswies/embedding_on_the_wall/tree/main/d_nos-nu-news-images-classification/datasets/images/svc_classification_corrected

The proposed method consists of three steps which are applied in sequence to create the visualization. The first step is the creation of a face detection model with one or more images as input. The output of such a model consists of images of all faces within the input images. Here different models are trained, tested and compared to the model used in Ruyters (2025). The best of these models is chosen as explained in section 4.1 and is used in the interactive visualization.

The second step is the creation of a face clustering algorithm which should be able to cluster all faces in a given set of faces such that every unique face has its own cluster. Within the final algorithm, the input images for this algorithm are the output images of the face detection model. Also, multiple cluster algorithms will be implemented of which the best algorithm as explained in section 4.2 is chosen to be used in the interactive visualization.

Finally, a face can be inserted into the visualization of which the user would want the results as explained in section 4.3. The model should then be able to apply face recognition and find the correct cluster that matches the input face if such a cluster exists. To do so, a simple face recognition will be fitted on the output of the clustering algorithm as explained in the paper of Ruyters, and then be used to predict the cluster for the input image.

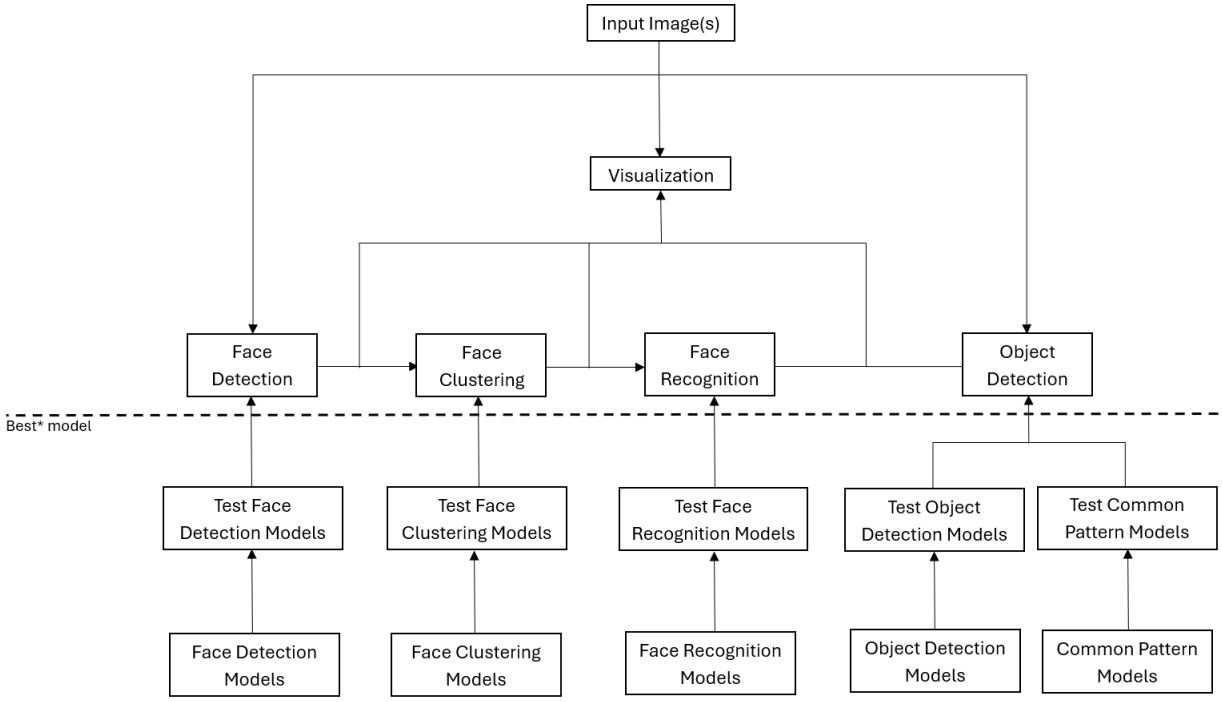


Figure 4: Sketch of approach (Not up to date: Object detection is removed, and Face Recognition has no best model selection.)

4.1 Face Detection

Since the basis of this thesis lies in the paper of Ruyters (2025), we should use that model as a base model to find improvements. Therefore, the base model for face detection consists of a combination of a face detection and face recognition model named face_recognition[6] and OpenCV's haar-cascade[7]. Face_recognition is a face recognition model based on dlib's face recognition model[8]. According to their package page, it has an accuracy of 99.38% on the Labelled Faces in the Wild dataset, but their model can also be used to detect faces. Haar-cascade is a model with which multiple objects can be detected like human bodies, faces and eyes. For face detection haar-cascade offers a few models which can be used, we will use the same model as Ruyters. By combining these models, more faces will be detected in an image and the detected faces will be more accurate to the actual faces.

One promising face detection model to improve on the base model is InsightFace. Insightface is an open-source deep face analysis library, offering multiple face recognition models. We will use the so-called buffalo_l model which uses the SCRFD-10GF architecture which has an accuracy of 95.16% (Guo et al., 2021).

In addition, the YOLO architecture is used. There are YOLO models from the internet that can be used for face detection for different versions of the YOLO architecture (Jocher et al., 2023). For this project, we use the oldest smallest model and the newest larger model, respectively YOLOv6n-face and YOLOv11l-face. Recently, a new version of YOLO came out which has not been implemented as a face recognition model yet[9]. Since training a model costs a significant amount of time, all versions of YOLOv12 (YOLOv12n, -s, -m, -l and -x) are compared with each other based on a smaller dataset, which is the FDD-WF small subset dataset.

To evaluate the performance of these models, the following evaluation metrics are stored: mAP50-95, mAP50, precision, recall, and runtime (training time). The difference between mAP50-95 and mAP50 is that mAP50 uses a threshold for the intersection over union (IoU) of 0.50, and mAP50-95 takes the average of different IoU thresholds between 0.50 and 0.95, the definition of the IoU will be explained later. Using these evaluation metrics on the validation set of the FDD-WF small subset dataset shows how well the model develops itself. Now, one model is chosen based on these metrics to be trained on a larger dataset which is the FDD-WF subset dataset. To train this model, similar parameters are passed through, as how the YOLOv6n-face - YOLOv11l-face models are trained.

For all models mentioned above, we used the same test data. Since the intended use for these models is the identification of faces in picture leads in newspapers considering politics, a representative dataset should be used. Therefore, the data scraped from the internet by Ruyters has been used and reduced to a set of 290 images such that every image contains some politician, this is the FD-test dataset. These images have been annotated by hand

[6] `https://github.com/ageitgey/face_recognition`

[7] `https://github.com/opencv/opencv/tree/master/data`

[8] `https://dlib.net/`

[9] Checked at 01-03-2025

as to what you should expect a face detection model to recognize. An example of this can be seen in Figure 5, the entire test set can be found at the Roboflow project (BEP, 2025).

Figure 5: Example image of annotated test data for face detection. Original image from: https://www.nos.nl/artikel/2489302

To choose the best model among those tested, various evaluation metrics are used. At first, the overall IoU is used. This calculates for each pixel, whether it is detected or predicted as a face, the formula for the IoU is shown in equation 1. Also, the accuracy is measured where the IoU must be over some threshold. This means that for each predicted face, the IoU with all actual faces is calculated and if it meets the threshold, the actual face can't be used again and it is considered to be a true positive. The thresholds for this are 0.5 and 0.7, the threshold of 0.5 is generally used as a base since it means that a face is detected but that the bounding box doesn't fit the actual bounding box well. The threshold of 0.7 is used because it allows to check for predicted bounding boxes that fit the actual bounding box better. Moreover, the latency is used to keep track of the speed of a model, this is the average time taken to detect faces on one image.

$$\text{IoU} = \frac{\#\text{pixels where both a face is predicted and it actually is (overlap)}}{\#\text{pixels where either a face is predicted or one actually is (union)}} \tag{1}$$

4.2 Face Clustering

To cluster faces, one must first create face embeddings on which a model or algorithm can afterward calculate the similarity and the according clusters. To accomplish this, Ruyters

made use of the FaceNet model to calculate those face embeddings and then created a pipeline to use for clustering. However, this pipeline has to be trained on some data that does not meet the requirements for the needed clustering model. This model must be able to, given a set of face embeddings, cluster faces where all faces can be faces never seen before. One of the models that is tested is the Chinese Whispers algorithm. For this algorithm, the main parameter is the threshold used for determining whether an edge is added to the network. The weight for the edge is the similarity which is determined by normalizing Euclidean distance between a pair of vertices in the network and subtracting it from one as shown in equation 2.

$$\text{Similarity}(i,j) = 1 - \frac{\sqrt{\sum_{x=1}^{n}(\text{embedding}[j]_x - \text{embedding}[i]_x)^2}}{max(\forall_{u,v}[u \neq v : \sqrt{\sum_{x=1}^{n}(\text{embedding}[v]_x - \text{embedding}[u]_x)^2}])} \quad (2)$$

Another method used for clustering is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm which has two important parameters that determine how well the algorithm performs. The epsilon parameter determines the maximum distance between a pair of points to be considered in the neighbourhood of each other and the min_samples parameter determines the number of points that must be a neighbour of an arbitrary point v, to be considered a core point. This algorithm only clusters points that are core points and if two core points are in the neighbourhood of each other they are considered to be within the same cluster. To find out how well both of these algorithms perform, it is important to try different parameters. Moreover, both the CFI dataset and the Embedding on the Wall dataset will be used to test the clustering algorithm to check how the algorithms perform under different circumstances. For the CFI dataset, the faces must be extracted first using the detection model that comes out best according to the metrics as explained in section 4.1.

Another important factor in clustering faces is how they are embedded since using different face embedding calculations can significantly affect the resulting clustering. Therefore this is also tested. The face_recognition package can, besides detecting faces, also calculate face embeddings based on a provided bounding box of a face. In addition, DeepFace has a feature to calculate face embeddings with different models, namely: VGG-Face, Facenet, OpenFace, DeepFace, Dlib, ArcFace (Serengil and Ozpinar, 2024, 2020).

To ensure the algorithms provide better solutions to the clustering problem, it is important to reduce the dimensionality of the face embeddings before applying the clustering algorithms (Paukkeri et al., 2011; Kaski, 1998). Besides it leading to a better clustering, it also leads to a more structured plot when plotting the clusters. Ruyters has shown that t-distributed Stochastic Neighbor Embedding (t-SNE) can be used to create a clear visualization where the clusters are more separable rather than using Principal Components Analysis (PCA) to reduce the embeddings' dimension as can be seen in Figure 13.

To evaluate the algorithms two evaluation metrics are used: the Adjusted Rand Index (ARI) and the relative number of clusters. The ARI-score is used to compute the similarity between two clusterings, thus in this case the resulting clusters from the clustering algorithms and the actual clusters. Besides, the ARI-score is adjusted to take into account,

that an algorithm can 'guess' a cluster correctly, the equation for the ARI-score is shown in equation 4. To calculate the relative number of clusters see equation 5

$$\text{RI} = \frac{\#\text{agreeing\_pairs}}{\#\text{pairs}} \tag{3}$$

$$\text{ARI} = \frac{(\text{RI} - \mathbb{E}[\text{RI}])}{(max(\text{RI}) - \mathbb{E}[\text{RI}])} \tag{4}$$

$$\text{rel} = \#\text{predicted\_clusters} - \#\text{actual\_clusters} \tag{5}$$

4.3 Interactive Visualization

To allow the user to have influence on the model and be able to understand how well the model functions, an interactive visualization is implemented. This visualization allows the user to see the results of the algorithm on the data scraped by Ruyters. Also, the user is allowed to bring their own set of images to run the algorithms on. To future-proof the visualization a bit, it is allowed to add some additional data along with the image in which a potential user might be interested in, think of categories of news article. A potential user for this interactive visualization would be a researcher who is in the possession of a set of images and wants to find co-occurring persons in those images. A full list of user tasks is shown in Table 1.

Level	**Action**	**Target**	**Task description (participant)**	**Realization**
Analyze	Enjoy	Features	Be exposed to all images uploaded by the user when creating a new database, in order to become aware of what the detection model lacks in	Allow user to scroll through all images with their detections
			Be exposed to all face embeddings when creating a new database and the clustering algorithm's parameters, in order to become aware of how these parameters affect the clustering	Allow user to play with the parameters and show them the resulting clustering
	Annotate	Features	Be able to add, remove and edit any faces in an image uploaded by the user when creating a new database	Interactive visualization
			Be able to add, remove and edit any cluster in the clustering when creating a new database	

Search	Lookup	Dependecy	Find out who co-occurs on an image in which the person of interest exists	Horizontal or vertical bar chart
			Find out what values of some additional data are related to the images which the person of interest (and a second selected person) is in	(ordered)
		Extremes	Find out which person co-occurs the most with the person of interest in a certain database	
			Find out which value, of all values of some additional data related to the images which the person of interest (and a second selected person) is in, is most common	
	Locate	Features	Find out whether a specific person of interest exists in the clustering of a certain database	Face recognition model on user input
Query	Identify	Features	Find out in what image the person of interest (together with a second selected person) exist in	Scroll through all images with selected person(s)

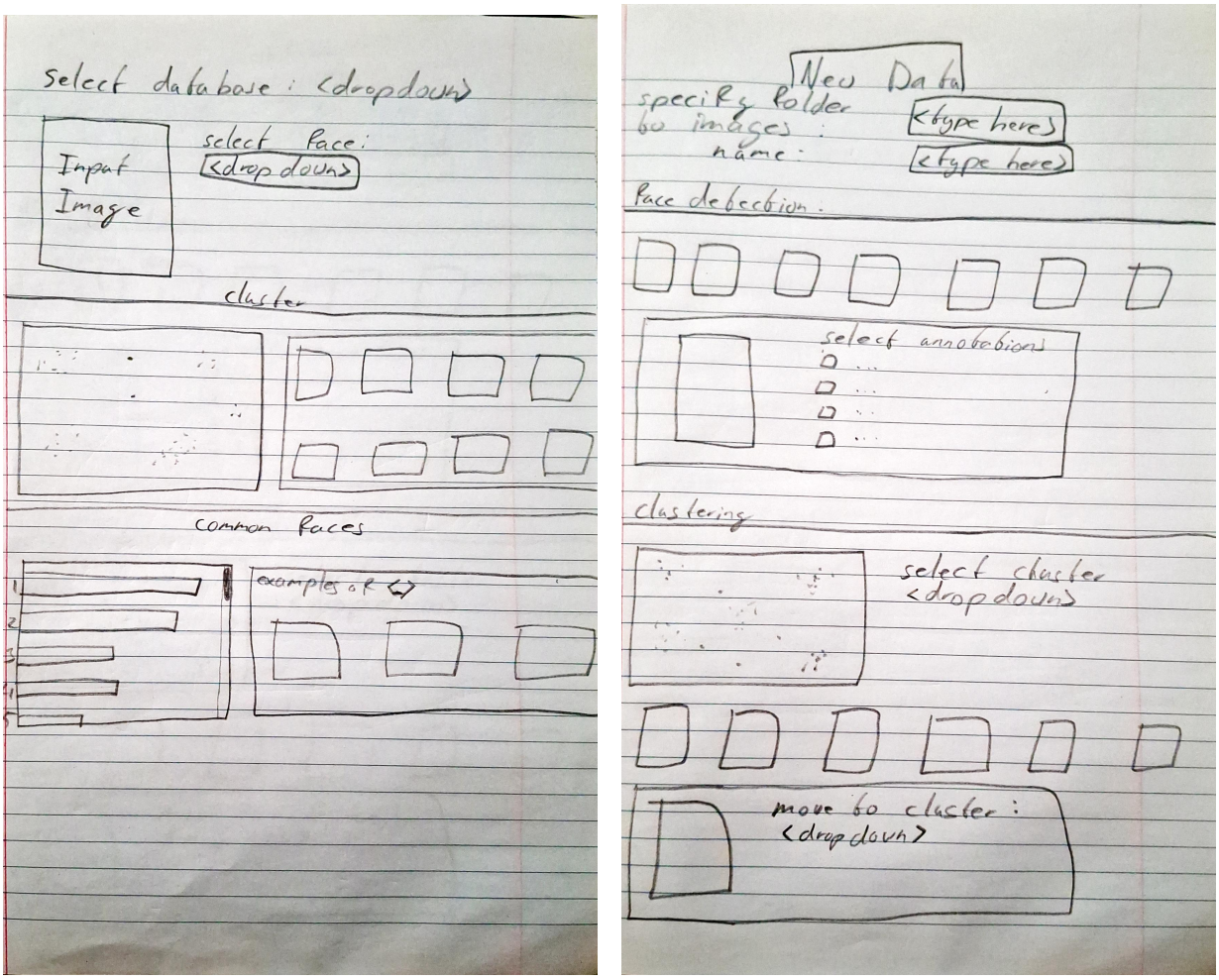
Table 1: The task abstraction for the interactive visualization, based on the methodology described in Munzner (2014)

In Figure 6a a sketch of the 'Visualization' page of the interactive visualization is described. Here the user is able to select one of the databases on which a simple face recognition model is trained as explained by Ruyters, and the user can input an image from their pc. After which they can select a face detected on the image or draw a bounding box themselves. The face recognition model then tries to predict the correct cluster based on the input image, an example of the predicted cluster is shown to check for the user manually. If the user already knows who he is interested in, he can select any cluster from a radio-menu instead of inputting an image. At first, the cluster in which the selected face is recognized will be highlighted by colouring this cluster red within the scatter plot where all other points are light-gray. Next to this plot, the user is able to see some example images from the highlighted cluster. Since there are most likely many more images than fit on your screen, two buttons are added to go through all images, this holds for all such example images sections in this visualization. In the section below the results are shown, a horizontal bar chart shows the user which faces are shown in the same image as the selected face, sorted from most common to least common to allow the user to easily see

the most common person. Here, the length of a bar encodes the number of times that person co-occurs with the person of interest in an image. Moreover, the user can click on any of these bars to see some example images where both the selected person (bar) and the person of interest occur in. To allow the user to find out where the images come from, they can click on any image to follow a URL if such a URL was provided. At the bottom, a horizontal bar chart is displayed where some additional information is shown, this is only shown if it exists in the database. Again, length encodes the number of times a certain value occurs in all images where the person of interest occurs in, and if selected, the person belonging to a selected bar occurs in. Also, the bars are sorted to easily allow the user to find out which value occurs most.

In Figure 6b a sketch of the 'New Data' page is described. This is where the user can upload a new dataset which can either be a set of one or more images, or a CSV-file where the user can input additional information and URL's. After all data is processed and the face detection model is done detecting all faces, the first few images are shown next to each other with the resulting bounding boxes plotted on them. The colour of the bounding boxes is chosen to be lime green since this is a distinctive colour for many different backgrounds. The user can select an image here and adjust which bounding boxes should be used, or add any extra bounding boxes. Also, the user can set a minimum size for faces to be considered for detection, to reduce the noise when clustering this set of faces. To address the problem of having many faces in an image, the user can show the face numbers to know which face to edit. After the user is satisfied with the detected faces, the user can continue to clustering all faces. Here, all clusters are shown in a scatter plot, using different unique colours for every cluster. Obviously, there can be many more clusters than unique colours distinguishable by a human, which is 12 colours according to Munzner (2014). however, if there are more clusters than colours, it leads to colours being reused. Therefore a larger colour set is needed and thus a set of 20 unique colours[10] is used. If a user is still unsure what colour belongs to what cluster, he can enable and disable the cluster in the legend of the scatterplot to see where the dots dis-/reappear. The user can now update the parameters for the clustering algorithm and try to fine-tune them. Also, the user is able to select any cluster after which some example images are shown from that specific cluster to allow the user to check whether the cluster meets their expectations. In case of any error in the clustering, the user is able to move an image to a different or new cluster. Besides, the user can draw an area in the scatterplot to create a new cluster, and the user is allowed to rename and merge clusters. If the user is satisfied with all the results from face detection as well as face clustering, the user can save the results to the database and use the visualization page to get the results on the new database.

[10] `https://cmap-docs.readthedocs.io/en/latest/catalog/qualitative/seaborn:tab20/`



(a) The visualization page

(b) The 'New Data' page

Figure 6: Sketch of the interactive visualization (Subject to change to a better looking one and add horizontal bar chart).

5 Results

5.1 Face Detection Model Selection

In Table 2, the results from all different YOLOv12 versions are shown. Here it is shown that YOLOv12s scores best on almost all evaluation metrics and has the lowest training time which is more ideal for training the final model since it requires more data and more epochs. Therefore, the final YOLO model is trained using the YOLOv12s architecture.

model version	YOLOv12n	YOLOv12s	YOLOv12m	YOLOv12l	YOLOv12x
mAP50	0.495	**0.506**	0.494	0.464	0.338
mAP50-95	0.265	**0.279**	0.267	0.249	0.180
precision	0.760	0.762	0.731	**0.770**	0.655
recall	0.433	**0.437**	0.433	0.390	0.290
runtime (mm:ss)	05:20	**05:02**	08:02	14:39	32:00

Table 2: Evaluation metrics for different versions of the YOLOv12 architecture trained using the FDD-WF small subset dataset.

Since the final YOLO model is trained for 100 epochs, there is a chance that it may overfit to the training data. To establish whether and when the model is overfitting, the losses are compared for the training and validation set (Figure 7). From the figure, it is clear that the validation loss stops reducing around epoch 60 and even starts to slowly increase later on while the training loss is still steadily decreasing. This is a sign of overfitting and therefore the model weights for epoch 60 are used as the final YOLOv12s-face version.

Figure 7: Dfl-loss comparison between the training and validation phase while training the YOLOv12s model.

The final results of the different face detection models are shown in Table 3. It is shown that for all models, there is a significant increase in the overall intersection over union and the accuracy compared with the base model. Among these models, YOLOv12s-face scores the best according to these evaluation metrics and YOLOv6n-face is the fastest. Since both models are fast and YOLOv12s-face only takes an extra 15 milliseconds per image, YOLOv12s-face is the best face detection model among the ones tested.

model	IoU (entire image)	Accuracy (IoU $\geq$ 0.5)	Accuracy (IoU $\geq$ 0.7)	Latency (s)
(base model)	0,473	0,561	0,224	0,455
InsightFace	0,821	0,877	0,775	0,086
YOLOv6n-face	0,830	0,875	0,770	**0,071**
YOLOv11l-face	0,825	0,870	0,766	0,222
YOLOv12s-face	**0,843**	**0,883**	**0,789**	0,086

Table 3: Evaluation metrics for face detection models tested using the FD-test dataset.

5.2 Face Clustering

In Figures 14-20 the t-SNE reduced embeddings are plotted for each model and colored as to the actual clusters. For both the CFI dataset and the Embedding on the Wall dataset, the faces embedded using DeepFace and OpenFace are all over the place as if the data would be considered noise. Since there are no visual clusters for these two models, it is not possible for a distance-based clustering algorithm to find clusters close to the actual clusters of these datasets. Therefore these two models will not be further considered for embedding faces. For all other models, there are visual clusters, so these models will be considered for the next step. But it must be noted that there always seems to be at least one cluster of noise which makes it harder for distance-based clustering algorithms to create clusters close to the actual clusters.

In Tables 5 & 6 the resulting evaluation metrics are shown for different values of input variable epsilon in the DBSCAN algorithm for both datasets. Both tables show that the ArcFace model and the Dlib model have a decent ARI score, but have significantly more predicted clusters than there actually are. For the CFI dataset, the FaceNet and face_recognition models both have high ARI scores while remaining close to the actual number of clusters. However, looking at the results using the Embedding on the Wall dataset, the embedding by FaceNet shows a larger difference in the number of clusters than there actually are, the same goes for the VGG_Face model. For both datasets, it holds that using the face_recognition model to embed faces leads to the clustering which is most similar to the actual clusters when using the DBSCAN algorithm. Moreover, it is notable that the range of optimal values for the epsilon parameter differs for both datasets.

In Tables 7 & 8 the resulting evaluation metrics are shown for different values of the threshold for the Chinese Whispers clustering algorithm for both datasets. For the CFI dataset, all models used for calculating the embeddings perform well with both a high ARI-score and with the predicted number of clusters being close to the actual number of clusters. The face_recognition model performs best for this dataset with the highest ARI-score. For the Embedding on the Wall dataset, the face_recognition model and VGG_Face model perform best, of which the face_recognition model results in the highest ARI-score. But for both models, the resulting number of clusters is further from the actual number of clusters than it was for the CFI dataset.

To compare the two algorithms with each other, the best results of both models are

shown in Table 4. Overall, the ARI-scores for both clustering algorithms are good, but DBSCAN has the overall highest scores. Moreover, the time to compute the clusters grows significantly more for the Chinese Whispers algorithm than the DBSCAN algorithm when the input size is increased.

	CFI dataset			Embedding on the Wall		
model	ARI	rel	latency (s)	ARI	rel	latency (s)
DBSCAN	0.950	1	0.060	0.964	0	0.022
Chinese Whispers	0.950	1	9.515	0.905	-3	0.626

Table 4: Results for both clustering algorithms in terms of evaluation metrics where ARI = adjusted rand score and rel = the difference in resulting and actual clusters.

5.3 Interactive Visualization

To develop a better understanding of the functionalities of the interactive visualization, a few use cases are provided.

Use Case 1
Suppose you are a researcher interested in the politicians represented in the Dutch news header images during the last elections. And you are especially interested in the people who are commonly shown in these header images together with an unknown person of which you have an image, and what kind of issues those articles discuss. Now, this interest overlaps with the interest of Ruyters, of which the database is already available when you navigate to the 'visualization' page and select the database 'Embedding_On_The_Wall_fixed.db'. Now, the user can select any person of interest from the radio menu, but this user does not know the name of the person. Therefore, the user can upload an image and select the person he is interested in. As can be seen in Figure 8, the YOLOv12s model found 3 faces in the uploaded image. To find out which face matches which name in the radio menu, the user can check the 'show face numbers' checkbox. When selecting a face in the radio menu, a simple face recognition model predicts that the face matches the face of Geert Wilders. The user can now check if the faces shown on the right-hand side match, and if they do not match, the user can select one of the clusters from the radio menu and try to find a matching cluster to the uploaded image. It can be the case that such a cluster does not exist in the database.

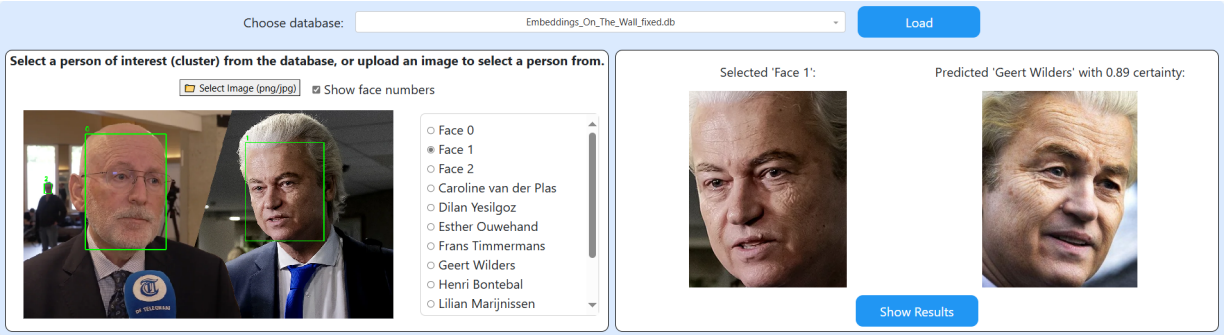


Figure 8: Section on the 'visualization' page where the user can select a person of interest.

Clicking on the 'Show Results' button, two new sections appear: the 'Explore Cluster' section and the 'Results' section. In the 'Explore Cluster' section the user can check whether all images indeed include Geert Wilders. In the 'Results' section, the user can find what he is interested in. Here the user finds all persons who are found in the same images as Geert Wilders, and by clicking on a person in the bar chart, the user can check all images to see if both these persons exist in all of them. Now, the user can select any additional information of his interest, which is 'Issues_paragraphs' in this case. Then, the user is able to see the kind of issues discussed in news articles related to the two selected persons as can be seen in Figure 9. Additionally, the user can click on any image on this page to navigate to the corresponding news article if needed.

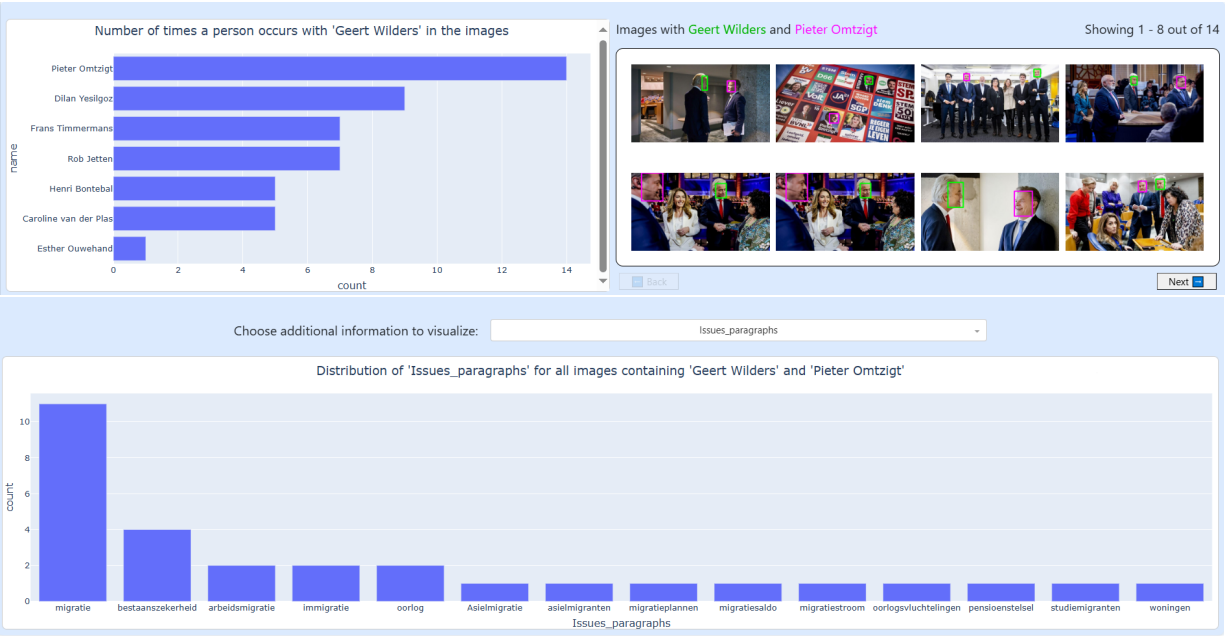


Figure 9: Results section on the 'visualization' page where the user can select another person who co-occurs in at least one image with the person of interest, check those images and see which issues are commonly discussed in those articles.

Use Case 2

Consider a user who is interested in celebrities and whether there are any celebrities who are commonly in the same images in a dataset the user gathered. Since there is no database by default in this visualization, the user must create one. this can be done when navigating to the 'New Data' page. Here the user uploads an arbitrary amount of images in jpg or png format from his dataset. Once the face detection model is done with detecting all faces a new section pops up where he can add and remove any face to the set of selected faces as can be seen in Figure 10.

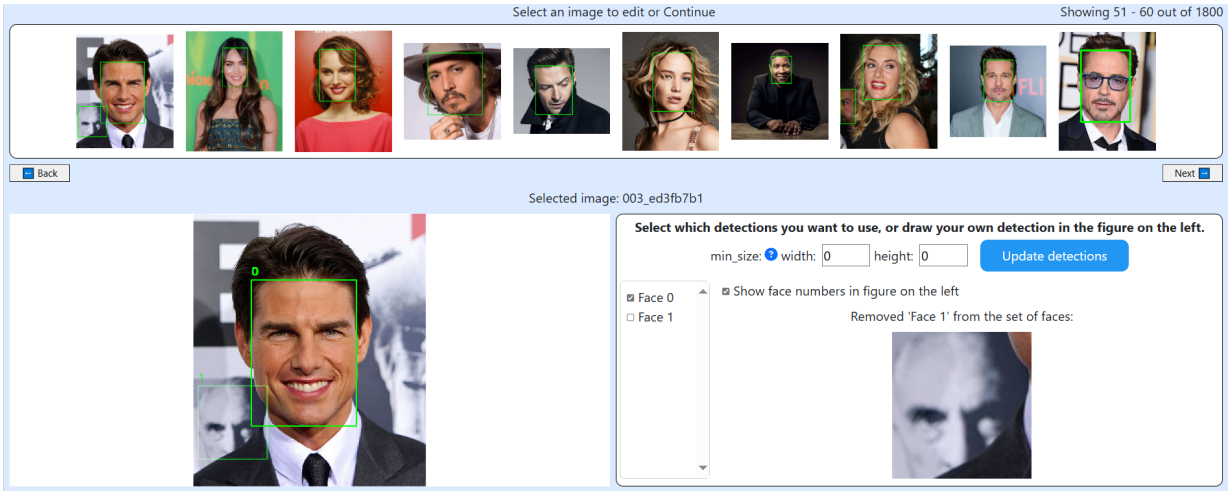


Figure 10: Face Detection section on the 'New Data' page where a user can add and remove any face from the selected set of faces. Also, any potential noise (background images) can be removed by setting a minimum size for faces.

Once the user is satisfied with the set of selected faces, he proceeds to the clustering section where the parameters for the DBSCAN clustering algorithm can be fine-tuned. The progression of this can be seen in Figure 11. Note that some faces have gone to the unknown class since they do not meet the requirement of having 8 neighbours within 2.6 distance.

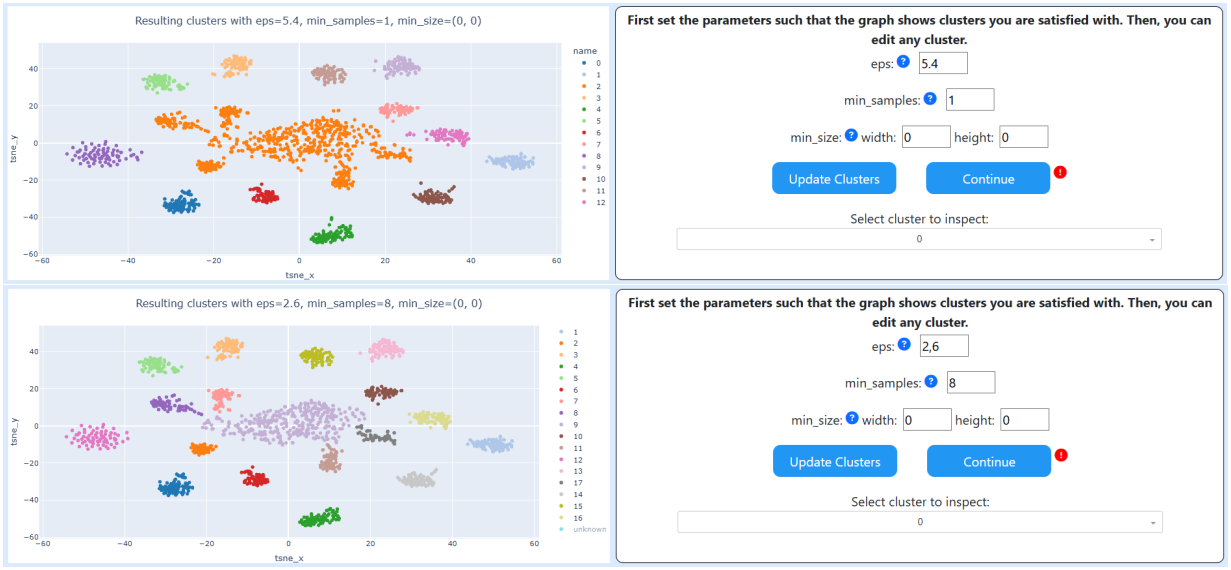


Figure 11: Face Clustering section on the 'New Data' page in the visualization, showing a dataset before (upper) and after (lower) fine-tuning the parameters for the DBSCAN clustering algorithm.

When the user finds values for the parameters satisfying their expectations, the user can start editing clusters by adding and merging clusters and by moving single images from one cluster to another. Moreover, the user can name the clusters to allow for easier interpretation when going through the results. This process can be seen in Figure 12.

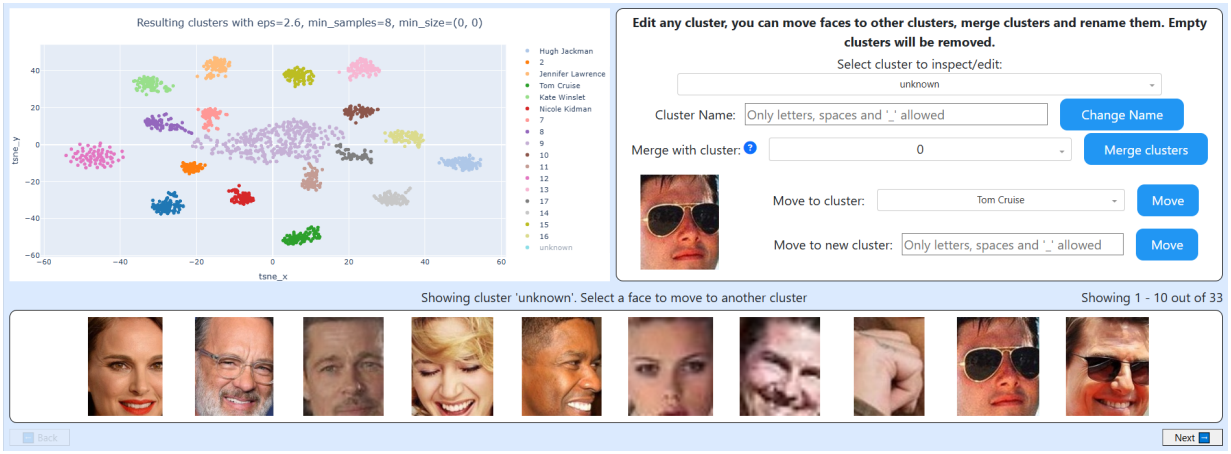


Figure 12: Face Clustering section on the 'New Data' page in the visualization, where a user can add, edit and remove any cluster to his liking.

When the user is done with the manual clustering, he can save the database with an inputted name. To see the results and check who are commonly in the same image, the user can follow similar steps as in use case 1 by selecting the recently saved database.

The difference is that the user now knows which person(s) he is interested in so instead of inputting an image he can just select a cluster from the radio menu. Also, since the user did only input images and not a CSV-file with additional data, no additional information as shown in the lower image in Figure 9 will be displayed.

6 Conclusion and Discussion

For the face detection, it is recommended to use the YOLOv12s-face model since it performs best among the models tested according to metrics used during evaluation. But there are still some problems with this model, sometimes the model predicts multiple faces on one face which have therefore overlapping bounding boxes. However, when two faces are behind each other, the model can predict both faces correctly but they still have overlapping boxes.

When clustering faces there are two possible algorithms tested in this thesis. Both algorithms perform best when used with faces embedded by the face_recognition model. Both models have high ARI-scores when used with the correct parameters, but finding the correct values for these parameters can be time-consuming. If you prefer an easy setup and do not care much about runtime, the Chinese Whispers algorithm is a great option. If the speed of the algorithm is of greater importance, the DBSCAN clustering algorithm is recommended. But there may still be better algorithms to test, think of algorithms like Clustering Using REpresentatives (CURE) (Guha et al., 2001).

Some problems with the visualization may occur when trying to cluster datasets that are small or have many background images or single-occurring faces in images. Both tested clustering algorithms are based on distances between two points and therefore lack the ability to handle noise correctly which then must be handled by the user manually. This is because the face embeddings do not guarantee that similar faces through the eyes of a human get similar embeddings. One solution to this could be to try other face embedding methods. But this can also be partially solved within the visualization if the user would be allowed to select some clusters that form unique faces, after which a face recognition model would be trained on that and try to predict all remaining faces to one of those clusters or an unknown cluster.

Also, to further expand the visualization, one could look further into the use of the additional information section as to what may be interesting to the user. Also, one could add extra models that may be in the user's interest, think of object detection models that detect certain objects or models that find similar patterns in a pair of images.

Another thing the visualization includes but should be improved on is the simple face recognition model used when identifying which cluster would belong to a certain inputted face. For now, the base model as used by Ruyters was applied, but it still lacks the ability the correctly identify someone consistently. In the sense that the certainty of a prediction may differ every time the prediction is made again. However, this can be due to the threshold which was set being too low or too high.

References

Amel Ibrahim Al Ali, Sheeja Rani S, and Ahmed M. Khedr. Enhancing financial distress prediction through integrated chinese whisper clustering and federated learning. *Journal of Open Innovation: Technology, Market, and Complexity*, 10(3):100344, 2024. ISSN 2199-8531. doi: 10.1016/j.joitmc.2024.100344.

Momina Liaqat Ali and Zhou Zhang. The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers*, 13(12), 2024. ISSN 2073-431X. doi: 10.3390/computers13120336. URL https://www.mdpi.com/2073-431X/13/12/336.

BEP. Face detection dataset. https://universe.roboflow.com/bep-jzm3u/face-detection-nmcfg, mar 2025. URL https://universe.roboflow.com/bep-jzm3u/face-detection-nmcfg. visited on 2025-04-03.

Chris Biemann. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In Rada Mihalcea and Dragomir Radev, editors, *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/W06-3812/.

Adam Geitgey. Face recognition: Recognize and manipulate faces from python or from the command line, 2020. URL https://github.com/ageitgey/facerecognition.

Scott Grissom, Myles F. McNally, and Tom Naps. Algorithm visualization in cs education: comparing levels of student engagement. In *Proceedings of the 2003 ACM Symposium on Software Visualization*, SoftVis '03, page 87–94, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136420. doi: 10.1145/774833.774846.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35–58, 2001. ISSN 0306-4379. doi: https://doi.org/10.1016/S0306-4379(01)00008-4. URL https://www.sciencedirect.com/science/article/pii/S0306437901000084.

Jia Guo, Jiankang Deng, Alexandros Lattas, and Stefanos Zafeiriou. Sample and computation redistribution for efficient face detection, 2021. URL https://arxiv.org/abs/2105.04714.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023. URL https://github.com/ultralytics/ultralytics.

S. Kaski. Dimensionality reduction by random mapping: fast similarity computation for clustering. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 1, pages 413–418 vol.1, 1998. doi: 10.1109/IJCNN.1998.682302.

Zengmin Lin, Chaoqun Hong, Weiwei Zhuang, and Keshou Wu. Face clustering based on fusion of face tracking and optimization. In *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, ICCPR '20, page 208–212, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450387835. doi: 10.1145/3436369.3437420.

Tamara Munzner. Visualization analysis and design, 2014.

T.L. Naps. Jhave: supporting algorithm visualization. *IEEE Computer Graphics and Applications*, 25(5):49–55, 2005. doi: 10.1109/MCG.2005.110.

Mari-Sanna Paukkeri, Ilkka Kivimäki, Santosh Tirunagari, Erkki Oja, and Timo Honkela. Effect of dimensionality reduction on different distance measures in document clustering. In Bao-Liang Lu, Liqing Zhang, and James Kwok, editors, *Neural Information Processing*, pages 167–176, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24965-5.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alina Petukhova, João P. Matos-Carvalho, and Nuno Fachada. Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108, December 2025. ISSN 2666-3074. doi: 10.1016/j.ijcce.2024.11.004. URL `http://dx.doi.org/10.1016/j.ijcce.2024.11.004`.

Wies Ruyters. Embedding, embedding on the wall; who is the most prominent politician of them all? exploring automated methods to study multi-modal political news coverage. 2025.

Purvi Saraiya, Clifford A. Shaffer, D. Scott McCrickard, and Chris North. Effective features of algorithm visualizations. *SIGCSE Bull.*, 36(1):382–386, March 2004. ISSN 0097-8418. doi: 10.1145/1028174.971432.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. URL `https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Schroff_FaceNet_A_Unified_2015_CVPR_paper.html`.

Sefik Ilkin Serengil and Alper Ozpinar. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 23–27. IEEE, 2020. doi: 10.1109/ASYU50717.2020.9259802. URL `https://ieeexplore.ieee.org/document/9259802`.

Sefik Ilkin Serengil and Alper Ozpinar. Hyperextended lightface: A facial attribute analysis framework. In *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–4, 2021. doi: 10.1109/ICEET53442.2021.9659697.

Sefik Ilkin Serengil and Alper Ozpinar. A benchmark of facial recognition pipelines and co-usability performances of modules. *Bilisim Teknolojileri Dergisi*, 17(2):95–107, 2024. doi: 10.17671/gazibtd.1399077. URL `https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077`.

Karen Shiells and Peter Pham. Unsupervised clustering for language identification, 2010. URL `https://snap.stanford.edu/class/cs224w-2010/proj2010/28_writeup.pdf`.

Siffi Singh and Hsueh-Ming Hang. A continuous learning system for face clustering and recognition. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2021. doi: 10.1109/ICCE50685.2021.9427771.

Jacob Solawetz. What is mean average precision (map) in object detection?, May 2020. URL `https://blog.roboflow.com/mean-average-precision/`.

A Appendix

A.1 Dimensionality Reduced Clustering

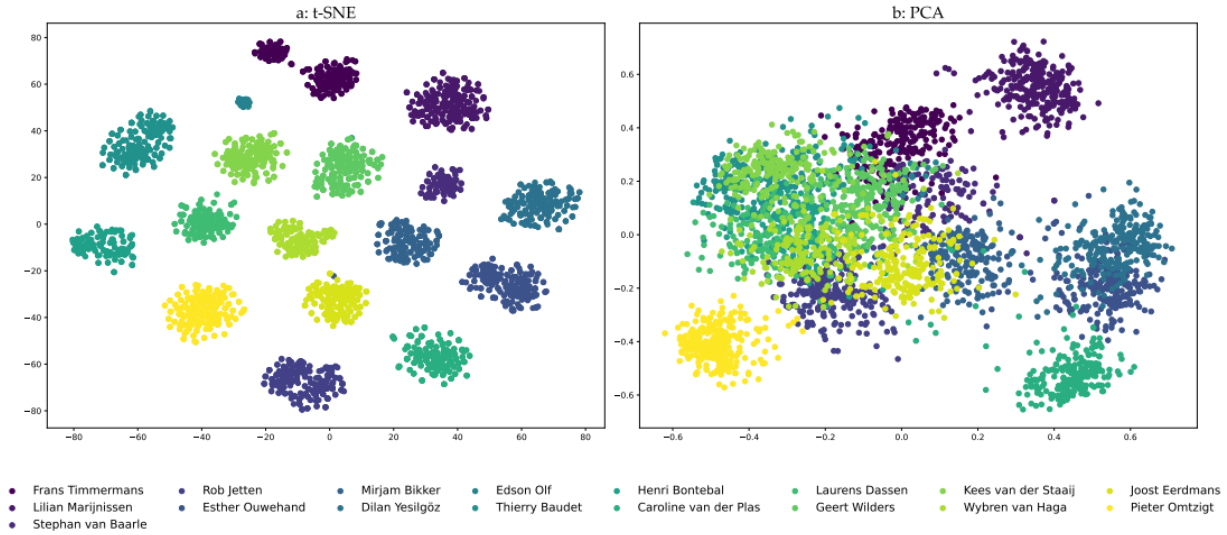


Figure 13: Visualization of embeddings clusters where they are reduced in dimensionality using t-SNE (a) and PCA (b) as shown in the paper from Ruyters.

A.2 t-SNE Reduced Embeddings with Different Models

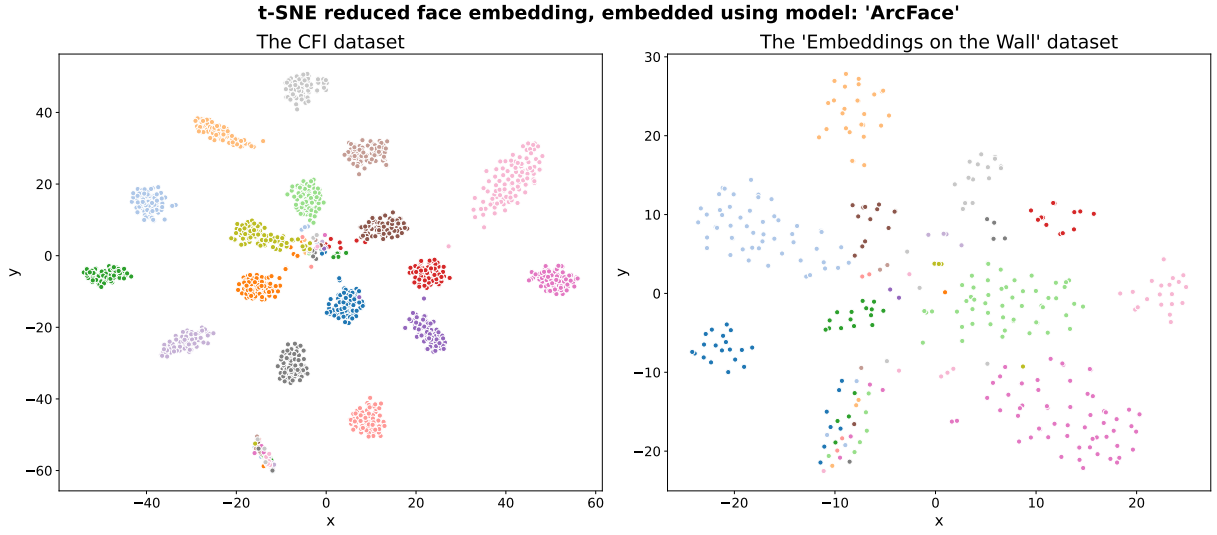


Figure 14: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the ArcFace model. Each color defines a unique person.

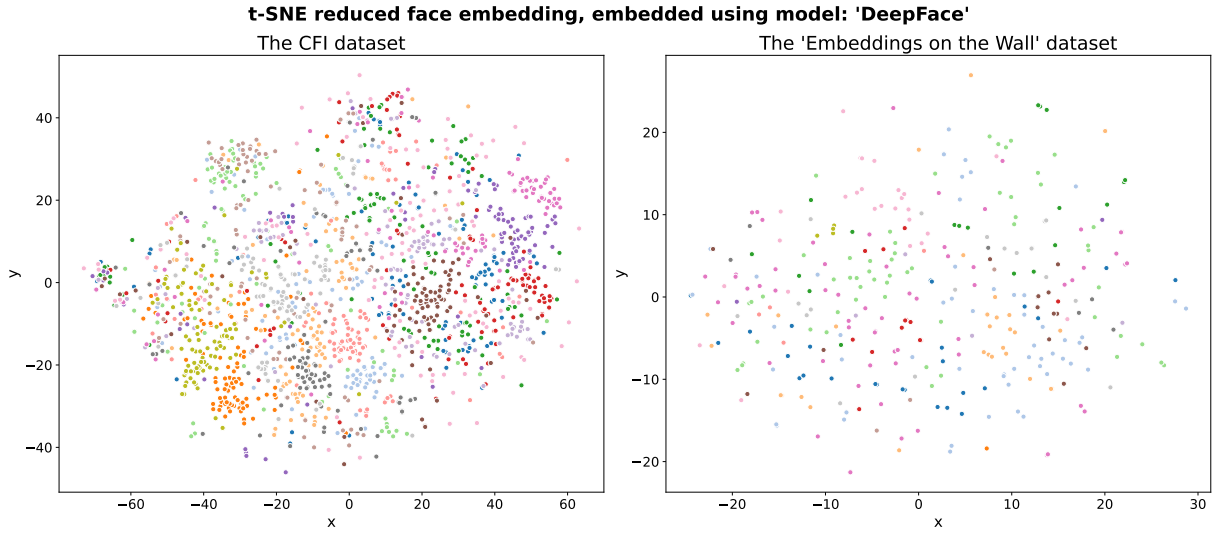


Figure 15: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the DeepFace model. Each color defines a unique person.

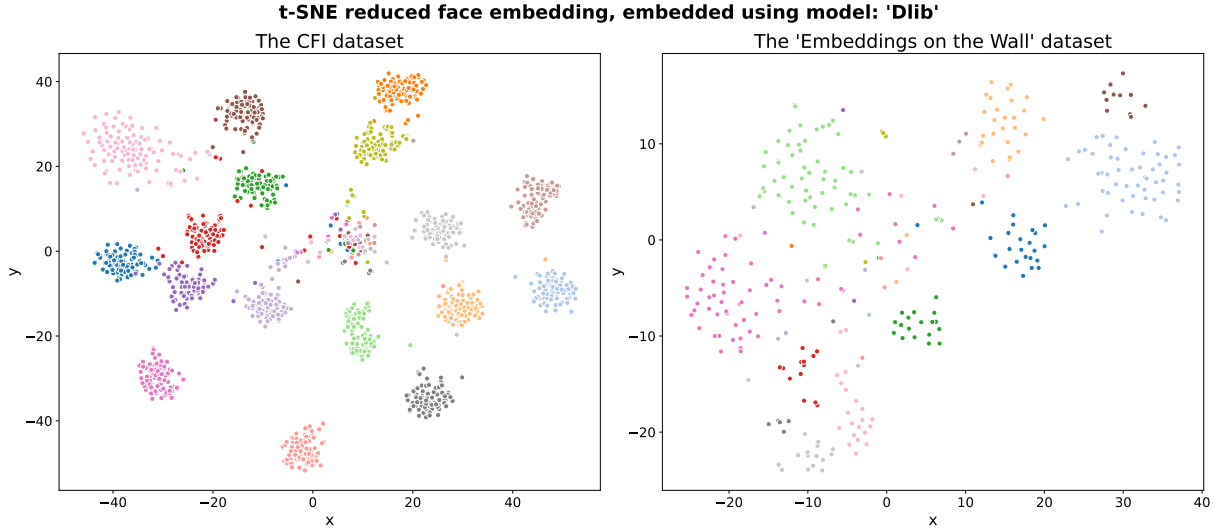


Figure 16: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the Dlib model. Each color defines a unique person.

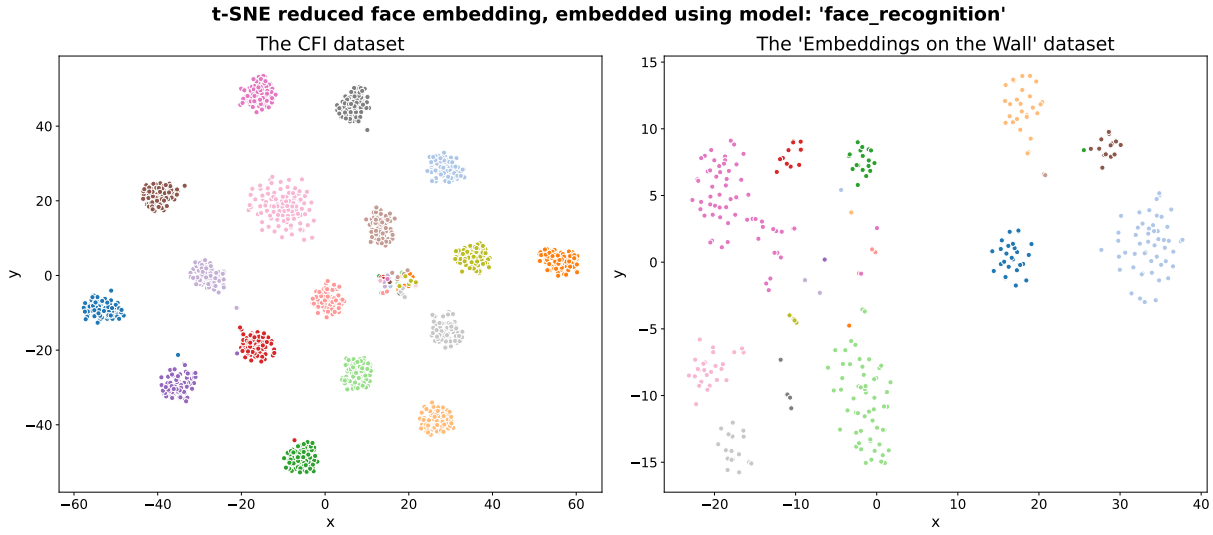


Figure 17: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the face_recognition model. Each color defines a unique person.

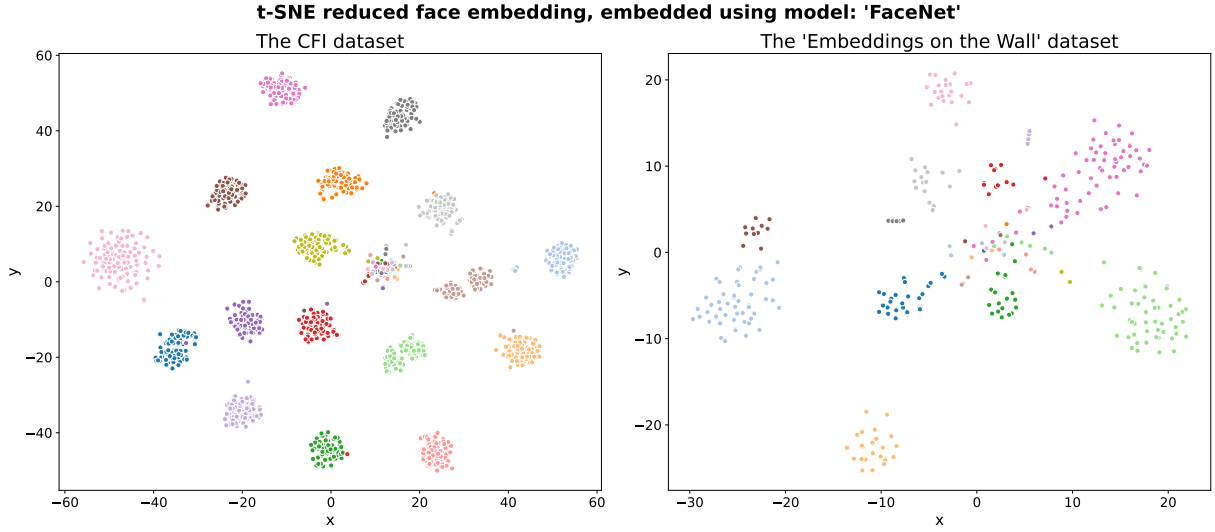


Figure 18: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the FaceNet model. Each color defines a unique person.

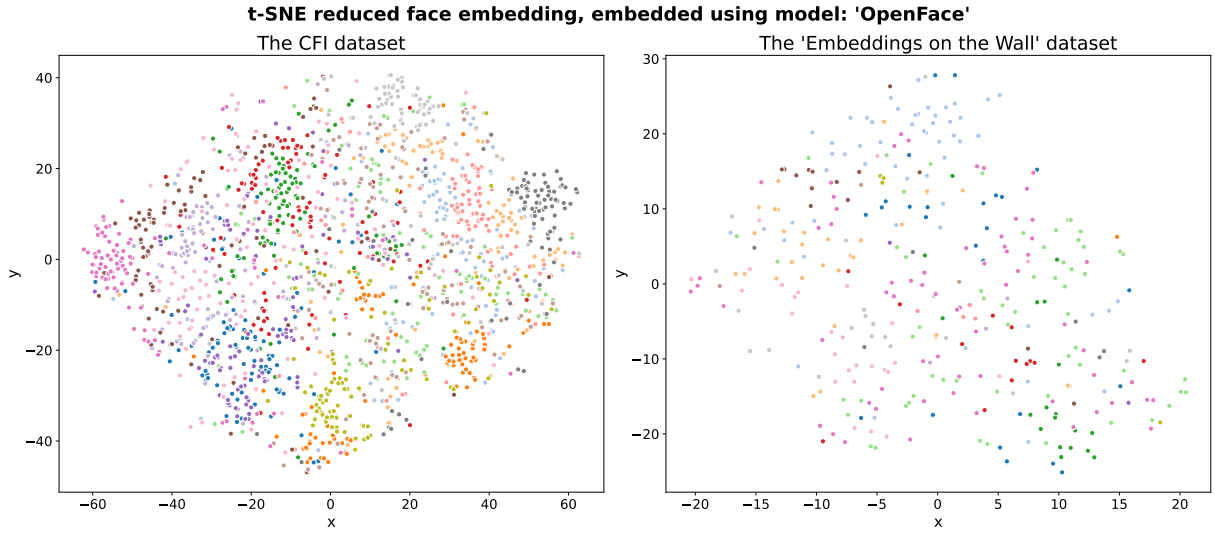


Figure 19: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the OpenFace model. Each color defines a unique person.

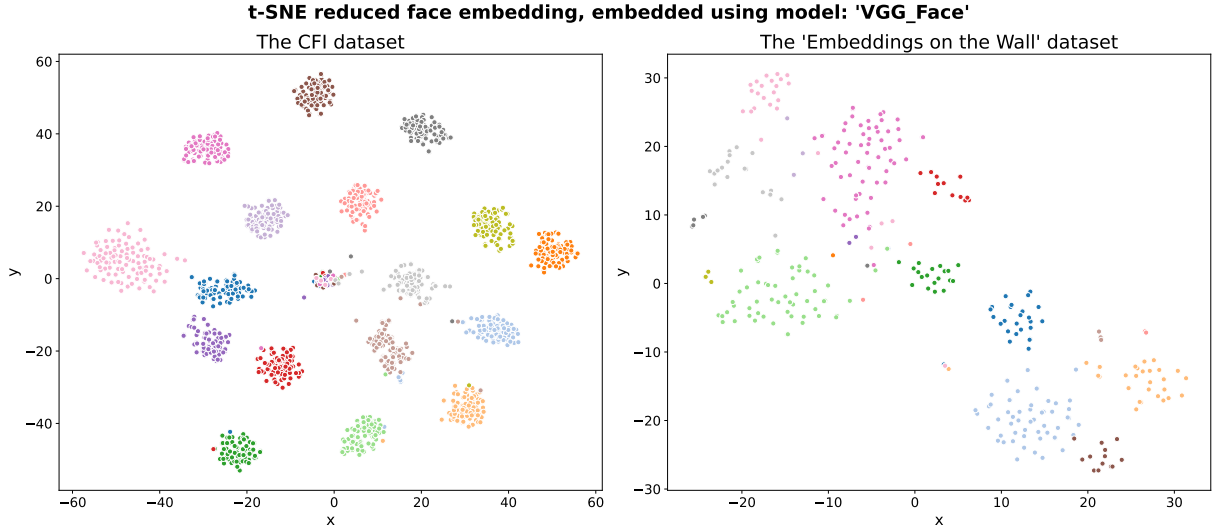


Figure 20: Actual cluster of all faces in both the CFI dataset (left) and the Embedding on the Wall dataset (right) embedded using the VGG_Face model. Each color defines a unique person.

A.3 Testing Parameters for Clustering with DBSCAN

params	ArcFace		Dlib		FaceNet		face_recognition		VGG_Face	
eps	ARI	rel	ARI	rel	ARI	rel	ARI	rel	ARI	rel
0.1	0.003	1616	0.003	1630	0.004	1588	0.003	1605	0.003	1599
0.2	0.006	1477	0.005	1521	0.007	1449	0.007	1441	0.006	1476
0.3	0.010	1347	0.008	1398	0.011	1289	0.012	1291	0.009	1359
0.4	0.015	1175	0.012	1249	0.020	1086	0.017	1136	0.014	1201
0.5	0.024	997	0.019	1098	0.039	876	0.030	943	0.021	1049
0.6	0.047	799	0.037	924	0.084	683	0.063	735	0.030	906
0.7	0.104	619	0.082	723	0.153	516	0.156	537	0.053	742
0.8	0.226	431	0.170	547	0.305	376	0.358	368	0.119	570
0.9	0.384	311	0.298	423	0.426	279	0.577	245	0.221	434
1.0	0.557	218	0.430	335	0.547	207	0.659	173	0.356	323
1.1	0.677	150	0.507	262	0.636	154	0.726	132	0.466	252
1.2	0.729	107	0.640	194	0.708	115	0.789	91	0.543	197
1.3	0.754	85	0.697	155	0.735	94	0.821	67	0.636	155
1.4	0.795	63	0.751	124	0.774	72	0.854	52	0.697	120
1.5	0.807	51	0.768	103	0.811	60	0.870	43	0.756	90
1.6	0.807	41	0.785	87	0.823	51	0.882	37	0.817	69
1.7	0.831	30	0.801	76	0.843	45	0.891	31	0.850	52
1.8	0.870	21	0.807	65	0.849	34	0.901	22	0.874	39
1.9	**0.874**	17	0.810	57	0.872	26	0.916	15	0.887	31
2.0	**0.874**	15	0.812	53	0.882	22	0.931	10	0.900	26
2.1	0.811	13	0.820	45	0.890	16	0.938	8	0.907	20
2.2	0.816	10	0.828	39	0.895	15	0.938	8	0.914	15
2.3	0.808	8	**0.830**	34	0.896	13	0.941	7	0.914	15
2.4	0.806	6	0.806	26	0.896	12	0.941	7	0.921	13
2.5	0.808	5	0.814	19	0.901	10	0.941	7	0.924	11
2.6	0.808	5	0.774	14	0.904	8	0.946	6	0.924	11
2.7	0.808	4	0.774	14	0.904	8	0.947	5	0.928	9
2.8	0.808	4	0.774	14	0.909	6	0.947	5	0.928	9
2.9	0.810	3	0.717	10	0.922	5	0.949	3	0.930	7
3.0	0.810	3	0.722	8	0.922	5	0.949	3	0.934	5
3.1	0.809	2	0.722	7	0.921	4	0.949	3	0.934	5
3.2	0.809	2	0.724	4	0.921	4	**0.950**	2	**0.939**	4
3.3	0.809	2	0.724	4	0.921	4	**0.950**	2	**0.939**	4
3.4	0.809	2	0.699	3	0.921	4	**0.950**	2	**0.939**	4
3.5	0.812	1	0.699	3	0.921	4	**0.950**	2	**0.939**	4
3.6	0.812	1	0.700	1	0.921	4	**0.950**	2	**0.939**	4
3.7	0.812	1	0.700	1	0.921	4	**0.950**	2	**0.939**	4
3.8	0.719	0	0.700	1	0.921	4	**0.950**	2	**0.939**	4

3.9	0.719	0	0.673	0	0.921	4	**0.950**	2	**0.939**	4
4.0	0.719	0	0.673	0	0.921	4	**0.950**	2	**0.939**	4
4.1	0.623	-1	0.673	0	0.921	4	**0.950**	2	**0.939**	4
4.2	0.623	-1	0.672	-1	0.921	4	**0.950**	2	0.894	3
4.3	0.623	-1	0.673	-2	0.922	3	**0.950**	2	0.894	3
4.4	0.623	-1	0.673	-2	**0.924**	2	**0.950**	2	0.894	3
4.5	0.623	-1	0.673	-2	**0.924**	2	**0.950**	2	0.894	2
4.6	0.623	-1	0.673	-2	**0.924**	2	**0.950**	2	0.894	2
4.7	0.623	-1	0.672	-3	**0.924**	2	**0.950**	2	0.894	2
4.8	0.604	-2	0.577	-4	**0.924**	2	**0.950**	2	0.894	2
4.9	0.603	-3	0.577	-4	**0.924**	2	**0.950**	2	0.747	0
5.0	0.603	-3	0.401	-5	0.886	1	**0.950**	2	0.747	0
5.1	0.603	-3	0.401	-5	0.886	1	**0.950**	2	0.747	0
5.2	0.603	-3	0.292	-6	0.886	1	**0.950**	2	0.747	0
5.3	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.747	0
5.4	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.747	0
5.5	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.747	0
5.6	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.747	0
5.7	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.714	-1
5.8	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.714	-1
5.9	0.603	-3	0.292	-6	0.883	0	**0.950**	1	0.715	-2
6.0	0.510	-4	0.292	-7	0.883	0	0.904	0	0.715	-2
6.1	0.510	-4	0.292	-7	0.883	0	0.904	0	0.715	-2
6.2	0.510	-4	0.292	-7	0.883	0	0.904	0	0.715	-2
6.3	0.510	-4	0.286	-8	0.883	0	0.904	0	0.715	-2
6.4	0.510	-4	0.286	-8	0.883	0	0.904	0	0.715	-2
6.5	0.510	-4	0.286	-8	0.883	0	0.904	0	0.715	-2
6.6	0.510	-4	0.286	-8	0.883	0	0.904	0	0.715	-2
6.7	0.510	-4	0.279	-9	0.883	0	0.904	0	0.685	-3
6.8	0.510	-4	0.279	-9	0.883	0	0.904	0	0.685	-3
6.9	0.510	-4	0.189	-10	0.847	-1	0.904	0	0.655	-4
7.0	0.510	-4	0.189	-10	0.847	-1	0.904	0	0.655	-4
7.1	0.510	-4	0.189	-10	0.847	-1	0.904	0	0.655	-4
7.2	0.510	-4	0.118	-11	0.847	-1	0.904	0	0.655	-4
7.3	0.510	-4	0.118	-11	0.847	-1	0.904	0	0.655	-4
7.4	0.510	-4	0.118	-11	0.847	-1	0.904	0	0.655	-4
7.5	0.510	-4	0.118	-11	0.847	-1	0.882	-1	0.655	-4
7.6	0.510	-4	0.067	-12	0.847	-1	0.882	-1	0.655	-4
7.7	0.510	-4	0.067	-12	0.847	-1	0.882	-1	0.596	-5
7.8	0.510	-4	0.067	-12	0.847	-1	0.882	-1	0.519	-6
7.9	0.510	-4	0.067	-12	0.847	-1	0.882	-1	0.459	-7
8.0	0.510	-4	0.067	-12	0.847	-1	0.882	-1	0.459	-7
8.1	0.510	-4	0.066	-13	0.847	-1	0.882	-1	0.459	-7

8.2	0.510	-4	0.028	-14	0.847	-1	0.824	-2	0.459	-7
8.3	0.510	-4	0.028	-14	0.847	-1	0.824	-2	0.459	-7
8.4	0.510	-4	0.028	-14	0.780	-2	0.824	-2	0.459	-7
8.5	0.510	-4	0.028	-14	0.623	-3	0.824	-2	0.459	-7
8.6	0.510	-4	0.028	-14	0.623	-3	0.824	-2	0.459	-7
8.7	0.370	-5	0.028	-14	0.602	-4	0.824	-2	0.459	-7
8.8	0.370	-5	0.028	-14	0.561	-5	0.824	-2	0.459	-7
8.9	0.370	-5	0.028	-14	0.561	-5	0.824	-2	0.459	-7
9.0	0.370	-5	0.028	-14	0.561	-5	0.824	-2	0.459	-7
9.1	0.370	-5	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.2	0.370	-5	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.3	0.370	-5	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.4	0.370	-5	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.5	0.370	-5	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.6	0.252	-6	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.7	0.252	-6	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.8	0.201	-7	0.028	-14	0.473	-6	0.741	-3	0.459	-7
9.9	0.201	-7	0.028	-14	0.200	-8	0.741	-3	0.459	-7
10.0	0.201	-7	0.028	-14	0.200	-8	0.741	-3	0.459	-7

Table 5: All tested values for the epsilon parameter for the DBSCAN algorithm using the CFI dataset where ARI = 'adjusted rand score' and rel = the relative number of clusters.

params	ArcFace		Dlib		FaceNet		face_recognition		VGG_Face	
eps	ARI	rel	ARI	rel	ARI	rel	ARI	rel	ARI	rel
0.1	0.005	326	0.004	329	0.006	320	0.007	315	0.006	318
0.2	0.008	312	0.006	318	0.010	305	0.013	293	0.009	307
0.3	0.010	302	0.009	307	0.012	295	0.019	275	0.011	296
0.4	0.011	297	0.010	301	0.015	284	0.022	261	0.013	290
0.5	0.014	290	0.012	294	0.023	262	0.030	238	0.015	281
0.6	0.015	285	0.014	288	0.030	242	0.043	211	0.019	267
0.7	0.019	272	0.016	277	0.039	217	0.083	172	0.022	256
0.8	0.023	261	0.020	264	0.063	185	0.118	147	0.024	248
0.9	0.027	246	0.027	244	0.098	153	0.242	108	0.036	222
1.0	0.036	226	0.039	222	0.207	123	0.371	82	0.054	186
1.1	0.049	199	0.053	205	0.283	91	0.495	49	0.079	162
1.2	0.070	174	0.096	166	0.373	76	0.671	37	0.110	142
1.3	0.101	142	0.163	141	0.488	59	0.750	27	0.169	121
1.4	0.139	114	0.215	117	0.539	44	0.782	20	0.235	100
1.5	0.198	93	0.272	90	0.675	29	0.820	15	0.327	76
1.6	0.325	73	0.427	67	0.699	23	0.904	11	0.383	64
1.7	0.386	61	0.522	51	0.725	19	0.925	8	0.507	51
1.8	0.479	46	0.593	42	0.761	12	0.934	7	0.570	44
1.9	0.527	37	0.671	31	**0.800**	6	0.934	7	0.653	36
2.0	0.572	30	0.694	27	0.793	3	0.934	7	0.686	30
2.1	0.662	20	0.743	23	0.672	-1	0.962	3	0.725	22
2.2	0.695	13	0.750	16	0.647	-4	**0.964**	1	0.791	15
2.3	0.736	9	0.750	16	0.647	-4	**0.964**	0	**0.844**	10
2.4	**0.756**	8	**0.759**	14	0.522	-5	**0.964**	0	0.743	7
2.5	0.709	4	0.734	7	0.503	-6	**0.964**	0	0.743	7
2.6	0.514	0	0.726	5	0.506	-7	0.956	-2	0.698	4
2.7	0.507	-2	0.658	1	0.506	-7	0.929	-3	0.668	-1
2.8	0.507	-2	0.633	0	0.506	-7	0.929	-4	0.668	-1
2.9	0.451	-4	0.514	-4	0.506	-7	0.929	-4	0.671	-2
3.0	0.451	-4	0.514	-4	0.499	-8	0.929	-4	0.671	-2
3.1	0.433	-5	0.508	-7	0.380	-9	0.929	-4	0.681	-3
3.2	0.353	-6	0.313	-9	0.380	-9	0.911	-5	0.408	-5
3.3	0.358	-7	0.249	-11	0.380	-9	0.849	-7	0.408	-5
3.4	0.179	-8	0.248	-12	0.361	-10	0.741	-9	0.400	-6
3.5	0.171	-9	0.232	-13	0.361	-10	0.741	-9	0.381	-7
3.6	0.171	-9	0.145	-14	0.368	-11	0.741	-9	0.366	-8
3.7	0.171	-9	0.145	-14	0.368	-11	0.741	-9	0.366	-8
3.8	0.171	-9	0.145	-14	0.352	-12	0.741	-9	0.364	-9
3.9	0.172	-10	0.004	-15	0.352	-12	0.690	-10	0.315	-10

4.0	0.138	-11	0.004	-15	0.352	-12	0.690	-10	0.315	-10
4.1	0.095	-12	0.004	-15	0.352	-12	0.690	-10	0.315	-10
4.2	0.097	-13	0.004	-15	0.352	-12	0.690	-10	0.313	-11
4.3	0.072	-14	0.004	-15	0.211	-13	0.654	-11	0.256	-12
4.4	0.072	-14	0.004	-15	0.211	-13	0.411	-12	0.256	-12
4.5	0.072	-14	0.004	-15	0.211	-13	0.411	-12	0.256	-12
4.6	0.072	-14	0.004	-15	0.211	-13	0.257	-13	0.212	-13
4.7	0.072	-14	0.004	-15	0.211	-13	0.257	-13	0.200	-14
4.8	0.072	-14	0.000	-16	0.211	-13	0.257	-13	0.200	-14
4.9	0.072	-14	0.000	-16	0.211	-13	0.257	-13	0.200	-14
5.0	0.072	-14	0.000	-16	0.211	-13	0.257	-13	0.200	-14
5.1	0.072	-14	0.000	-16	0.211	-13	0.257	-13	0.200	-14
5.2	0.072	-14	0.000	-16	0.211	-13	0.225	-14	0.200	-14
5.3	0.024	-15	0.000	-16	0.211	-13	0.192	-15	0.200	-14
5.4	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.200	-14
5.5	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.200	-14
5.6	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.200	-14
5.7	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.200	-14
5.8	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.200	-14
5.9	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.001	-15
6.0	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.001	-15
6.1	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.001	-15
6.2	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.001	-15
6.3	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.4	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.5	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.6	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.7	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.8	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
6.9	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.0	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.1	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.2	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.3	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.4	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.5	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.6	0.024	-15	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.7	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.8	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
7.9	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.0	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.1	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.2	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16

8.3	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.4	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.5	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.6	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.7	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.8	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
8.9	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.0	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.1	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.2	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.3	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.4	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.5	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.6	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.7	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.8	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
9.9	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16
10.0	0.000	-16	0.000	-16	0.153	-14	0.192	-15	0.000	-16

Table 6: All tested values for the epsilon parameter for DBSCAN using the Embedding on the Wall dataset where ARI = 'adjusted rand score' and rel = the relative number of clusters.

A.4 Testing Parameters for Clustering with Chinese Whispers

params	ArcFace		Dlib		FaceNet		face_recognition		VGG_Face	
threshold	ARI	rel	ARI	rel	ARI	rel	ARI	rel	ARI	rel
0.70	0.000	-16	0.092	-14	0.260	-13	0.000	-16	0.169	-14
0.71	0.000	-16	0.000	-16	0.105	-14	0.000	-16	0.170	-14
0.72	0.041	-15	0.102	-15	0.217	-13	0.000	-16	0.216	-14
0.73	0.000	-16	0.096	-14	0.000	-16	0.093	-15	0.218	-14
0.74	0.000	-16	0.102	-15	0.105	-14	0.116	-15	0.104	-14
0.75	0.041	-15	0.000	-16	0.136	-13	0.176	-14	0.209	-12
0.76	0.124	-12	0.204	-12	0.136	-13	0.029	-15	0.135	-14
0.77	0.220	-13	0.145	-13	0.136	-13	0.199	-13	0.271	-13
0.78	0.134	-12	0.316	-11	0.143	-12	0.204	-12	0.431	-10
0.79	0.134	-12	0.241	-11	0.143	-12	0.387	-10	0.252	-11
0.80	0.185	-10	0.193	-11	0.332	-10	0.256	-11	0.413	-9
0.81	0.403	-8	0.408	-9	0.462	-8	0.385	-9	0.428	-8
0.82	0.224	-8	0.245	-11	0.445	-7	0.603	-6	0.511	-7
0.83	0.277	-7	0.400	-9	0.749	-3	0.698	-4	0.628	-5
0.84	0.297	-6	0.526	-7	0.708	-3	0.674	-3	0.705	-4
0.85	0.368	-5	0.650	-3	0.812	-2	0.774	-2	0.697	-3
0.96	0.599	-3	0.650	-3	0.890	0	0.862	-1	0.850	-1
0.87	0.694	-2	0.671	-3	0.841	-1	0.863	-1	0.847	-1
0.88	0.793	0	0.730	-2	0.886	0	0.925	0	0.895	0
0.89	0.805	0	**0.862**	1	**0.927**	1	**0.950**	1	0.894	0
0.90	**0.902**	2	0.831	0	0.923	1	**0.950**	1	**0.939**	1
0.91	0.901	2	**0.862**	1	0.922	1	0.949	1	**0.939**	1
0.92	0.898	2	0.858	1	0.922	1	0.949	1	**0.939**	1
0.93	0.898	2	0.852	1	0.922	1	**0.950**	1	**0.939**	1
0.94	0.900	2	0.846	3	0.910	2	0.949	1	**0.939**	1
0.95	0.900	2	0.833	6	0.911	3	**0.950**	1	0.938	2
0.96	0.810	7	0.833	11	0.897	4	**0.950**	2	0.900	5
0.97	0.739	15	0.799	24	0.850	10	0.935	4	0.791	14
0.98	0.528	52	0.488	98	0.569	53	0.660	36	0.444	89
0.99	0.128	334	0.096	471	0.146	310	0.169	257	0.104	401

Table 7: All tested values for the threshold used for the Chinese Whispers clustering algorithm using the CFI dataset where ARI = 'adjusted rand score' and rel = the relative number of clusters.

params	ArcFace		Dlib		FaceNet		face_recognition		VGG_Face	
threshold	ARI	rel	ARI	rel	ARI	rel	ARI	rel	ARI	rel
0.70	0.000	-16	0.184	-15	0.153	-14	0.192	-15	0.199	-15
0.71	0.000	-16	0.184	-15	0.153	-14	0.192	-15	0.199	-15
0.72	0.000	-16	0.184	-15	0.153	-14	0.192	-15	0.199	-15
0.73	0.187	-15	0.234	-14	0.153	-14	0.192	-15	0.186	-15
0.74	0.000	-16	0.181	-15	0.265	-13	0.192	-15	0.199	-15
0.75	0.127	-15	0.234	-14	0.265	-13	0.192	-15	0.199	-15
0.76	0.217	-14	0.181	-15	0.265	-13	0.382	-14	0.376	-14
0.77	0.056	-14	0.232	-14	0.265	-13	0.355	-13	0.199	-15
0.78	0.121	-13	0.181	-15	0.430	-12	0.427	-13	0.386	-14
0.79	0.298	-12	0.232	-14	0.423	-12	0.427	-13	0.351	-14
0.80	0.162	-12	0.321	-14	0.276	-13	0.335	-14	0.386	-14
0.81	0.315	-11	0.331	-13	0.467	-11	0.484	-13	0.386	-14
0.82	0.475	-10	0.400	-13	0.449	-11	0.427	-13	0.345	-14
0.83	0.414	-10	0.374	-13	0.467	-11	0.619	-12	0.346	-14
0.84	0.419	-10	0.375	-12	0.460	-11	0.655	-11	0.516	-12
0.85	0.668	-8	0.640	-10	0.586	-10	0.745	-10	0.549	-11
0.86	0.551	-9	0.639	-10	0.668	-9	0.742	-10	0.736	-9
0.87	0.697	-7	0.617	-10	0.665	-9	0.745	-10	0.637	-10
0.88	0.693	-7	0.615	-10	0.729	-8	0.745	-10	0.743	-9
0.89	**0.745**	-6	0.618	-10	0.733	-8	0.775	-9	0.707	-10
0.90	0.744	-6	0.616	-10	0.707	-7	0.771	-9	0.764	-7
0.91	0.668	-3	0.707	-6	**0.792**	-6	0.856	-6	**0.812**	-4
0.92	0.679	-1	0.769	-3	0.767	-4	0.885	-4	**0.812**	-3
0.93	0.658	4	**0.770**	-2	0.752	-1	**0.905**	-3	0.773	-4
0.94	0.549	12	0.699	0	0.753	1	0.897	-3	0.731	2
0.95	0.319	26	0.483	15	0.726	5	0.797	1	0.707	9
0.96	0.219	48	0.424	35	0.421	23	0.721	8	0.429	24
0.97	0.126	100	0.271	61	0.267	59	0.531	24	0.265	57
0.98	0.045	199	0.082	160	0.101	137	0.243	71	0.096	131
0.99	0.014	288	0.014	285	0.023	260	0.044	205	0.022	256

Table 8: All tested values for the threshold used for the Chinese Whispers clustering algorithm using the Embedding on the Wall dataset where ARI = 'adjusted rand score' and rel = the relative number of clusters.