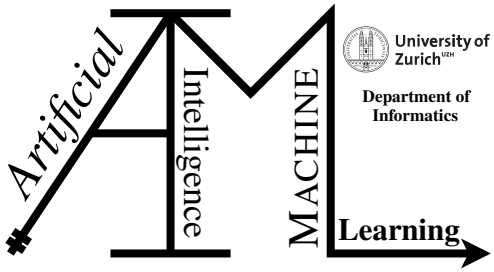
# Stability Analysis of Facial Attributes for Improved Face Recognition

Bachelor Thesis

**Ramize Abdili**
17-731-076

**Submitted on**
April 5 2021

**Thesis Supervisor**
Prof. Dr. Manuel Günther



**Bachelor Thesis**

**Author:** Ramize Abdili, ramize.abdili@uzh.ch

**Project period:** September 5 2020 - April 5 2021

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

# Acknowledgements

It is said far too seldom: I would like to express my deepest gratitude to Prof. Dr. Manuel Günther with whom it was and will be a pleasure to work. Thank you for the great collaboration, interesting discussions and insightful advice over the past six months. I really appreciate your competence and honesty very much, and I was always happy to have a look at my thesis with you.

Many thanks also go to the Artificial Intelligence and Machine Learning Group of the Department of Informatics of the University of Zurich for giving me the opportunity to work on such an interesting topic.

As well as my family, friends and all who have supported and encouraged me during this time, I would like to express my great praise and thanks.

# Abstract

In this thesis we do face verification, where the main task is to decide if two images belong to the same person or to different people. To do that we use the images from the Labeled Faces in the Wild (LFW) database. The images are taken in an uncontrolled environment and thus have natural variation in pose, lighting and background. We approach face verification through comparing facial attributes of two images from LFW. These facial attributes are extracted from our AFFACT network. To compare the facial attributes, we develop algorithms that perform different distance-based calculations between them. Some of our algorithms train a Support Vector Machine (SVM). After the comparison of two images, a score value is computed which deduces whether they belong to the same person and thus solve the face verification problem.

# Zusammenfassung

In dieser Arbeit werden wir eine Face Verification machen. Die Hauptaufgabe besteht darin, zu entscheiden, ob zwei Bilder derselben Person oder verschiedenen Personen gehören. Dazu verwenden wir die Bilder aus der Labeled Faces in the Wild (LFW) Datenbank. Die Bilder werden in einer unkontrollierten Umgebung aufgenommen und weisen daher natürliche Variationen in Bezug auf Pose, Beleuchtung und Hintergrund auf. Wir nähern uns der Face Verification, indem wir die Gesichtsattribute zweier Bilder von LFW vergleichen. Diese Gesichtsattribute werden aus unserem AFFACT-Netzwerk extrahiert. Um die Gesichtsattribute zu vergleichen, entwickeln wir Algorithmen, die unterschiedliche entfernungsbasierte Berechnungen zwischen ihnen durchführen. Einige unserer Algorithmen trainieren eine Support Vector Maschine (SVM). Nach dem Vergleich zweier Bilder wird ein Bewertungswert berechnet, der ableitet, ob sie derselben Person gehören, und somit das Problem der Face Verification löst.

# Contents

# Chapter 1

# Introduction

We live in a world where technology is evolving every day and playing an increasingly important role in our everyday life. The technology to unlock our smartphone is present in almost all new smartphones. This is just an example of where we find face recognition in our daily life. There are many other areas where this technique is frequently used such as surveillance and security systems. The advancement in deep learning in recent years has helped face recognition to become more accurate. Especially Deep Convolutional Neural Networks (DCNNs) have helped achieve very promising results.

As summarized by Guo and Zhang (2019) Deep Neural Networks (DNNs) consist of an input and output layers and many hidden layers in between them, which model complex relationships between the input and output. Convolutional Neural Networks (CNNs) are the most popular DNN used for face recognition as they achieve good results for image recognition. DCNNs have therefore become the standard for face recognition tasks. A typical CNN is built of convolutional, pooling and fully connected layers. The convolutional layers extract the features from the input data. The generated feature map of a convolutional layer is then passed to the next layer. This is done for each convolutional layer. The pooling layers are used for down-sampling. There are various DCNN methods for face recognition using a single CNN or multiple CNNs combined.

Face recognition is a process typically composed of multiple steps: face detection, face alignment, feature extraction and face recognition (Guo and Zhang, 2019).

As described by Kortli et al. (2020) the first step is face detection, where a whole scene image is given as input to the system and the faces are located and marked as bounding boxes. Face detection has greatly improved in speed and accuracy due to the DCNNs. The detection of a front facing person is no longer challenging, however the detection in more unconstrained environments can still produce some challenges. Typically, but not required, a next step is the alignment of the faces. After that, the features are extracted. This is usually done by choosing an DCNN architecture and a loss function for classification. A common way is to use transfer learning, where an DCNN is trained on a limited set of images and then an DCNN can be used as a descriptor extractor on unseen faces. As a last step the system compares the extracted features with the gallery faces to see if they are matching (Guo and Zhang, 2019). In face matching we differentiate between two tasks: face verification and face identification. Face verification aims at comparing two images and deciding whether they belong to the same person, whereas face identification is considered a closed-set problem because the assumption is that a query person is already in the gallery (Guo and Zhang, 2019).

One possibility how a machine learning technique can be trained based on a set of images is that a pair of two images of the same person or from different people is given as input. The technique is informed whether these pictures are of the same person or not. Then it extracts the features of them. In this feature extraction step, the facial attributes are extracted for each image. Such facial attributes are for example: the beard, the wearing of glasses, the pale skin, the thick

lips, the gender etc (Günther et al., 2017). After training the technique can predict whether a newly transferred pair of images is coming from the same person or not. This technique can therefore make predictions about whether a person is really who they claim to be.

Many large data sets for face recognition helped to advance this technology further. They allow the deep learning algorithm to train on a large training set and make more accurate predictions. Such data sets are for example the Labeled Faces in the Wild (LFW).[1] Despite the rapid improvement in performance and accuracy, especially for constrained images, face recognition for unconstrained images still remains a challenge. Since pictures are taken in an uncontrolled environment, they vary in condition such as lighting, head-pose, view-angle, background and camera quality (Huang et al., 2008). This variation between the images is a hurdle that limits the prediction of images whether they belong to the same person. For this reason, the task of this thesis is to develop algorithms which, despite this variation of images, can achieve the best possible prediction whether two images are of the same person. The pictures that we use are the pictures from LFW.

This work is organized as follows. Section 2 gives an introduction on what others have done in this field and how it relates to this work. In Section 3 it is described how the pictures in the LFW database are constructed. Section 4 explains how we use these pictures and how we develop the algorithms. In Section 5 the results of the algorithms are shown. The limitations that could have an influence on our work are discussed in Section 6. Section 7 discusses what possible work can still be done in the future. Finally, Section 8 gives the conclusion.

[1] http://vis-www.cs.umass.edu/lfw/

# Chapter 2

# Related work

According to authors Günther et al. (2017) and Rudd et al. (2016) the first task of face recognition is to detect the faces on pictures or videos. The second step is to decide whether the detected face is the face with the claimed identity. In order to perform face recognition, the features are extracted from images and compared with one another.

Kumar et al. (2009) have introduced three classifiers: the attribute classifier, the simile classifier and the verification classifier. The attribute classifier is trained to recognise if a facial attribute such as having a moustache or wearing glasses is present on a picture. When comparing images later to predict whether people in two images belong to the same person or not, the attribute classifier is not particularly helpful because it does not have to mean that two people who have many facial attributes in common are the same person. So, the second method is needed: the simile classifier. This classifier uses a set of 60 people and compares each attribute of a picture with the corresponding attribute of these 60 people and assigns to which of these 60 people it is most similar. For example, the attribute "Mouth" is similar to person number x in the set and the attribute "Eyes" is similar to person number y in the set. This is done for both faces which are compared to analyse how many attributes of both faces are assigned to the same person in the set to conclude how similar they are. The last method is the verification classifier which does mathematical calculations. All 73 facial attributes of one image are stored in a vector which looks like this:

$$\vec{v} = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_{73} \end{pmatrix}.$$

The calculation between two vectors of two images that want to be compared is done like this:

$$\vec{w} = \begin{pmatrix} \left| a_1^{(1)} - a_1^{(2)} \right| \\ \left| a_2^{(1)} - a_2^{(2)} \right| \\ \dots \\ \left| a_{73}^{(1)} - a_{73}^{(2)} \right| \\ a_1^{(1)} \times a_1^{(2)} \\ a_2^{(1)} \times a_2^{(2)} \\ \dots \\ a_{73}^{(1)} \times a_{73}^{(2)} \end{pmatrix}$$

where for example $a_2^{(1)}$ is the second attribute of person 1. All new calculated vectors $\vec{w}$ are put into the so called Support Vector Machine (SVM) together with the information whether the two

vectors, from which the new vector $\vec{w}$ was calculated, came from the same person or not. This information is required so that an SVM can be trained. Trained means that it can learn from the previous input to make decision in the future on how similar two new images will be.

Liu et al. (2015) have trained several Neural Networks to extract the facial attributes of an image in a feature vector. Here they have divided the attributes into two groups. Attributes such as gender or race are assigned into the identity-related attributes group and attributes such as wearing a hat or sunglasses into the identity-non-related attributes group. The performance of Neural Networks was better on the identity-related attributes group because these attributes are less sensitive to intrapersonal face variation. Also here the attribute vectors extracted by Liu et al. (2015) are used to train an SVM for attribute classification.

Zhang et al. (2014) have trained an CNN to recognise human attributes. These human attributes are based not just on a person's face, but on the entire human body such as clothing style. This enabled Zhang et al. (2014) to compare people by also comparing body structure and dress style. For training an CNN they have used poselets to align the human face part and the human body part, and to identify the pose and angle of view of faces in images. At the end, an CNN can detect whether a person has a human attribute or not.

Similar to Kumar et al. (2009), the Mixed Objective Optimization Network MOON architecture defined by Rudd et al. (2016) also have done a classification on the facial attributes. MOON has used a single DCNN to predict 40 facial attributes simultaneously, compared to Kumar et al. (2009) who have used 73 attributes. Before MOON, each attribute was classified individually by separate classifiers. MOON has achieved a multi-task learning and has done an optimization over all facial attributes at once rather than with every single one. Optimizing over all attributes at once reduces the error rate compared to optimizing each one individually. MOON is trained with the Euclidean loss function on pictures whose face attributes are labelled with binary numbers to predict them over a continuous range. A continuous range is advantageous for the representation of some attributes as it not only says whether a facial attribute is present or not, but also how strong it is. Rudd et al. (2016) have used only aligned pictures with faces rotated upright to train the network. They have conducted an experiment by deliberately misaligning the images by rotating and scaling them and have shown that misalignment is a problem for MOON because it was only trained with aligned pictures.

Günther et al. (2017) have described that MOON has used pictures from the CelebA database. CelebA database contains 200'000 images of 20'000 celebrities, 16'000 of them are assigned into the training set, 2'000 in the validation and the remaining 2'000 into the test set. Each picture has hand-labelled coordinates of the eyes, of the two corners of the mouth and of the nose tip. It also has its own vector with 40 attributes, in which its 40 attributes are represented as binary based on the presence or absence of the facial feature. The 40 facial attributes are shown in Figure 2.1. Based on the hand-labelled landmarks a bounding box will be calculated. This bounding box is used to crop the pictures so that only the person's face is included. At the end pictures are scaled so that they are all the same size. Additionally, a rotation angle is calculated by which the bounding box is rotated so that the faces are upright (Huang et al., 2007). Similar to Kumar et al. (2009) also here the vector $\vec{w}$ between two vectors is calculated. The vector contains 40 instead of 73 attributes and is put into an SVM to train it. Experiments have shown that the 40 attributes from MOON have a better performance in face recognition than the 73 attributes defined by Kumar et al. (2009) (Rudd et al., 2016).

Similar to MOON, the Alignment-Free Facial Attribute Classification Technique (AFFACT) defined by Günther et al. (2017) also has used an DCNN to perform the classification of all facial attributes and is trained with the Euclidean function on pictures whose face attributes are labelled with binary numbers to predict them over a continuous range. The main difference is that the images that AFFACT is trained on are free of alignment. Even if faces in pictures are upright, they are deliberately rotated so that they are no longer upright. Non-alignment or false alignment is

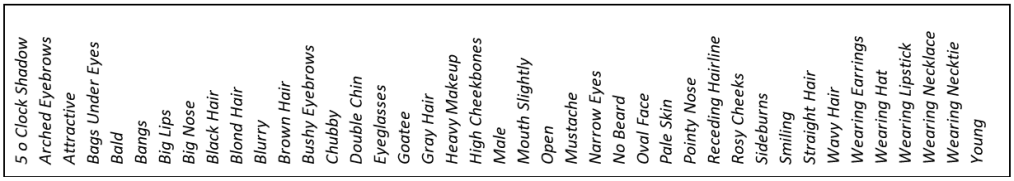



Figure 2.1: The 40 Attributes (Günther et al., 2017)

done on purpose so that the network can learn to work with crooked and misaligned faces and to deal with them in the future. AFFACT also has used pictures from the CelebA database but the step where the face is rotated is omitted. AFFACT's network trained on images with only one face bounding box based on the hand-labelled facial landmarks is shown to perform better than MOON's network trained on aligned images. The reason for this is that AFFACT's network was trained with images without alignment and can therefore handle localization errors very well.

Parkhi et al. (2015) have used an CNN. Each image from a data set is represented as vectors. During the training phase of an CNN the triplet-loss function is used. Triplets in form (a, p, n) are formed such that the vector of images a and p are from the same person and vector of image n is from another person and are the input data of an CNN. After that for the face verification the Euclidean distance between vectors of two images of the triplet is computed. An CNN can learn from this and remember how large the Euclidean distance is between images of the same person and from different people. In the test phase, the Euclidean distance between two images is also calculated. They are then classified as images from the same person if their distance is smaller than a threshold otherwise as images from different people. The threshold is defined in such a way that it maximizes the verification accuracy.

Wen et al. (2016) also have trained an CNN but with a center loss function. After extracting the features of images, for all features of one image the center loss function computes a centre. The goal is to minimize the distance between each feature to its centre so that the intra-class variation between features is minimized and inter-class variation is maximized. The intra-class variation is the difference in images of one person due to variation in pose, illumination and expression whereas the inter-class variation is the difference in images of two people (Sao and Yegnanarayana, 2007). After the function has done this for all images, the features of images are separated into clusters based on which centre they belong. Every time when a new feature is added to a cluster, the centre is recalculated by averaging all its features. To decide if two images are the same person or not, an CNN is trained by computing the Cosine similarity between two features.

Deng et al. (2019) also have used an CNN that extracts images as features and have the same goal, namely, to minimize the intra-class and to maximize the inter-class variation. They have created a loss function called Additive Angular Margin Loss ArcFace. With the Cosine function the angle between each two features is computed instead of computing a distance. Then an CNN is trained with the results of this Cosine function.

In this thesis we use a network that was trained with AFFACT data augmentation and the MOON technique and extract 40 continuous attributes for each image from the LFW database such that the attributes are classified. The attributes are balanced so that the error on the positive side equals the error on the negative side. A negative value of the attribute tells that this attribute is not presented in the image and a positive value tells the opposite. The range shows how strongly an attribute is present: the higher the value, the stronger. For the face verification we also compare two vectors of two images by calculating a distance between them to solve the problem of whether or not two images are the same person.

# Chapter 3

# Data

We use the database Labeled Faces in the Wild (LFW)[1] to deal with the unconstrained face verification problem. Some pictures from the database are shown in Figure 3.1.

As described by Huang et al. (2008) it contains 13'233 face pictures of 5'749 people of different ethnicity, gender, age, etc. 1'680 people have at least two pictures and the remaining 4'069 have just one picture. More information on how many people have how many pictures can be found in Table 3.1. All of them are hand-labelled and have a unique name. Most of them are coloured, only some are in grayscale. What can not be avoided is that the pictures show a natural variation in pose, lighting, expression, focus, resolution, facial expression, age, gender, race, ethnicity, clothing, hairstyles, accessories, make-up, occlusions, colour saturation, camera quality and background. Because of this unavoidable variation these images in the LFW database are called unconstrained face images.

Figure 3.1: Some Images of LFW (Huang et al., 2008)

| # of images /person | # of people (% of people) | # of images (% of images) |
|---|---|---|
| 1 | 4069 (70.8) | 4096 (30.7) |
| 2-5 | 1369 (23.8) | 3739 (28.3) |
| 6-10 | 168 (2.92) | 1251 (9.45) |
| 11-20 | 86 (1.50) | 1251 (9.45) |
| 21-30 | 25 (0.43) | 613 (4.63) |
| 31-80 | 27 (0.47) | 1170 (8.84) |
| $> 81$ | 5 (0.09) | 1140 (8.61) |
| **Total** | 5749 | 13233 |

Table 3.1: Distribution of LFW (Huang et al., 2008)

[1] http://vis-www.cs.umass.edu/lfw/

## 3.1 Construction of LFW

In this section we explain how the images in the LFW database are collected and how this database is constructed. All subsequent steps are described by Huang et al. (2008).

### 3.1.1 Gathering Raw Images

The first step is to collect a series of pictures. Berg et al. (2004) carried out this step by bringing together many raw images in the database.

### 3.1.2 Face Detection

After images are available in the database, they are individually scanned running a face detector called Viola-Jones. If a picture does not include a person or a face, this picture will be recognized by this detector and will not be added to the database. The detector also checks whether a certain image is already available in the database. If this is the case, it discards the image and thus eliminates duplicate images.

### 3.1.3 Labelling

As mentioned, all pictures with a detected face are hand-labelled with the unique name of that person. As an aid one could take the caption associated with an image. However, it is possible that certain people are given incorrect or more than one name. If one person received more than one name, then only the name more commonly seen is used. After the labelling, all pictures from one person are combined into the same subgroup.

### 3.1.4 Cropping and Re-Scaling

After running the face detector, it returns for each image a bounding box which only includes the face of that person. There is no information about the eye coordinates (Ruiz-del Solar et al., 2009). The centre is taken from the bounding box and the distance from the centre to the bounding box is measured. The bounding box is then enlarged by the factor 2.2 on all four sides and based on it the images are cut. After cropping out the images it can be that some images are very small or large compared to all of them together. So, they need to be re-scaled. The images from LFW we use are re-scaled to the size of 256x256 pixels and are then saved in the JPG format.

### 3.1.5 Forming Pairs of Training and Testing Sets

The database is randomly partitioned into two sets: the training data set contains 2'200 pairs; the test data set consists of the other 1'000 pairs. People who appear in the training data set do not appear in the test data set, and vice versa. Because of this separation experiments can be run several times without biasing the result. Based on the captions on the pictures, all pictures of one person are in a separate set. Then images in the training data set are used to create two further sets: the matched and mismatched pair sets. The matched pair set is created by first randomly picking a person who is present in at least two pictures. Then two pictures from the whole set of pictures are also randomly selected and added to the matching pair set, unless the pictures are the same picture, or the pair of pictures is already in the set. The mismatched pair set is created differently. First two random people are picked, no matter how many pictures they have. Then

a picture of each of them is selected and added as a pair in the mismatched pair set. Here, too, attention is paid to whether the pair of images is already present in the set. The same is done with the pictures from the test data set. In Figure 3.2(a) you can see some matched pairs and in Figure 3.2(b) some mismatched pairs of LFW. At the end there are 1'100 matched and 1'100 mismatched pairs in the training data set and 500 matched and 500 mismatched pairs in the test data set.

(a) Some Matched Pairs of LFW

(b) Some Mismatched Pairs of LFW

Figure 3.2: Some Matched and Mismatched Pairs of LFW (Huang et al., 2008)

## 3.2 LFW for the Unconstrained Face Verification Problem

As described by Huang et al. (2008) the decision whether two images are the same person or not is called the unconstrained face verification problem. The data from LFW are arranged into two views. View 1 is there to develop the algorithm and carry out experiments. As described in Section 3.1.5 the data is split into training and test data sets. In this View, researchers can use the images from the training set to train the algorithm and the test set for testing it and running experiments as often as they want by changing the parameters in the algorithm. View 2 reports the performance and is just used once at the end for the evaluation. The performance selected from the View 1 as the best of all algorithms is measured in View 2. View 2 has ten subsets: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10). Nine of them are the training set and one is the test set. The experimenter runs

ten experiments. In the first experiment the subsets with number (2, 3, .. 10) are used for training and the subset with number 1 for testing, in the second subsets (1, 3, 4, .. 10) are used for training and subset 2 for testing and so on. With this approach the experiments are run independently of one another. At the end, a mean accuracy and a standard error of the mean SE will be calculated. The mean accuracy is

$$\mu = \frac{\sum_{i=1}^{10} p_i}{10}$$

where i is the number of the subset which is used as the test subset and $p_i$ is the percentage of the correct classifications on this View. The standard error of the mean is

$$SE = \frac{\sigma}{\sqrt{10}}$$

with

$$\sigma = \sqrt{\frac{\sum_{i=1}^{10}(p_i - \mu)^2}{9}}.$$

Summarized View 1 is used for algorithm development to train and test many experiments with changing parameters and then the best experiment with the best performance is chosen. And View 2 is used for performance reporting and ten experiments with different defined training and test sets are run and then from all ten experiments the mean accuracy and the standard error of the mean are calculated. At the end you should describe which method was used for the training data set. These methods are described in Section 3.3.

## 3.3 Methods for Using the Training Data Set

Huang et al. (2008) have defined two main training methods: the image-restricted training method and the unrestricted training method. In the image-restricted training method the labels of the pair images in the training data set are not used to form more pairs in that set. For example if the pair images ($img_1$, $img_2$) and ($img_3$, $img_4$) of one person are in the training data set, then the images such as $img_1$ and $img_3$ are not considered that they come from the same person and are therefore not included as a pair in the training data set. This method just knows whether two images in a pair are matched and belong to the same person or are mismatched and do not belong to the same person. Based on the information if pair images are matched or mismatched this method can form new pair images and add them in the training data set. For example if the pair images ($img_1$, $img_2$) and ($img_3$, $img_4$) from one person are matched pairs in the training set then images such as $img_1$ and $img_3$ are seen as a matched pair as well and are added as a matched pair to the training data set. The unrestricted training method uses the labels of the pictures to know their identity and then creates a lot of new pairs as matched and mismatched pairs. To be able to do this, subsets of people were formed to help to create the new pairs. In this thesis we use the unrestricted training method.

According to authors Seo and Milanfar (2011) it is possible that experimenters do not need to use the training data set and create one themselves from scratch. This is an additional method called the unsupervised (no training) method. Since it is forbidden to use training examples, this method is the most difficult of all. The experimenters then have to calculate a similarity between images in order to identify them as matched and mismatched pairs. Such a calculation can be the Chi-square distance method.

## 3.4 Role of LFW in DAR-Pipeline

As discussed by Huang et al. (2008) with the Detection-Alignment-Recognition (DAR) pipeline, the faces are automatically detected, aligned and recognized. This means that faces can be detected not only on images but also on videos. The pipeline with its three steps is shown in Figure 3.3. Each step gets pictures of the previous step or forwards them to the next step after these pictures have been prepared. The images from the LFW database are prepared such that they have already gone through the detection step and can be used in this pipeline. Since the images in the LFW database only include the face, these faces can then be aligned and recognised.

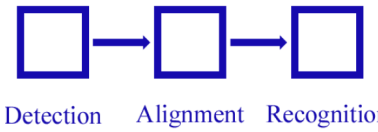



Figure 3.3: The Detection-Alignment-Recognition Pipeline (Huang et al., 2008)

(a) Images with their Marked Landmarks (b) Images with their Bounding Box

Figure 3.4: Images with their Marked Landmarks and Bounding Box (Huang et al., 2007)

### 3.4.1 Alignment

The use of the LFW images gives the opportunity to align the faces instead of having to worry about detecting the faces (Huang et al., 2008). This step is very helpful to improve the performance for the recognition step and makes the later face identification easier (Huang et al., 2007,

2012). Images can be aligned using the Zhou face alignment, which marks facial landmarks such as the coordinates of the eyes, the corners of the mouth and the nose tip on the images (Huang et al., 2007). This is shown in Figure 3.4(a). These facial landmarks are then used to define the bounding box, which is shown in Figure 3.4(b) (Huang et al., 2007). This bounding box is then rotated so that the face is put upright and based on it, the pictures are cut.

### 3.4.2 Recognition

After the alignment the faces are recognized (Huang et al., 2008). Thanks to the face detector and the alignment step, the faces were detected and correctly aligned so that it is easier to recognize a face in order to identify it (Günther et al., 2017). This means the better the face detection and alignment was, the better the identification will be. How exactly face recognition is made in our work is explained in Section 4.

# Chapter 4

# Methodology

In this section we describe how the pictures from the LFW database are used to resolve the unconstrained face verification problem. It has to be decided whether two faces from two pictures in the LFW database represent the same person or not (Huang et al., 2008).

As described on the following website[1] this problem is solved with the verification system, in which the features of both images are compared, and it is decided if they belong to the same person. This system is shown in Figure 4.1. In addition to this problem, there is another problem called the identification problem. Here by comparing two images, one is referred to as the "model" and the other as the "probe". Then the model image of the so-called enrolled template or gallery with the highest similarity is assigned to the probe image. The identification problem is solved with the identification system, which is shown in Figure 4.2. In this system the features of the probe image are compared to the features of all images in the gallery and the system picks out a model image from the gallery which best matches and resembles the probe image. As can be seen from Figures 4.1 and 4.2, both systems use the same steps: the pre-processor, feature extractor, matcher and decision maker, which are described below. What they also have in common is that the images are enrolled in the same way. This enrollment is marked with blue arrows on both figures. The only difference between them is the output. The verification system shows as output if two pictures are matched and the identification system assigns one image from the gallery which has the highest similarity to the probe image.

These two problems show two different ways to evaluate face recognition systems. Since the LFW database resolves just the verification problem and not the identification problem, we focus on the verification problem. We do not have a predefined gallery and define the first image of a pair from the test data set as the model image and the second as the probe image. The framework we use for this thesis is the bob.bio framework.[2] In order to resolve the verification problem, the images from the LFW database have to go through a few steps that will make their use and the decision easier.

[1] `https://www.idiap.ch/software/bob/docs/bob/docs/stable/bob/bob.bio.base/doc/`
[2] `https://www.idiap.ch/software/bob`

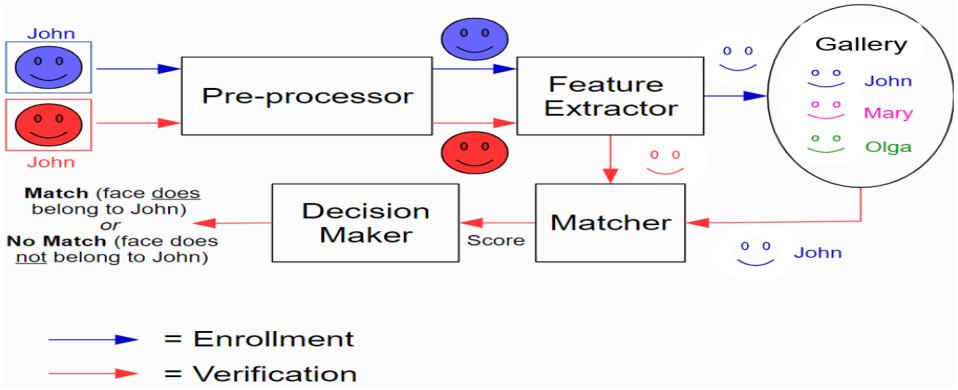



Figure 4.1: The Verification System[1]

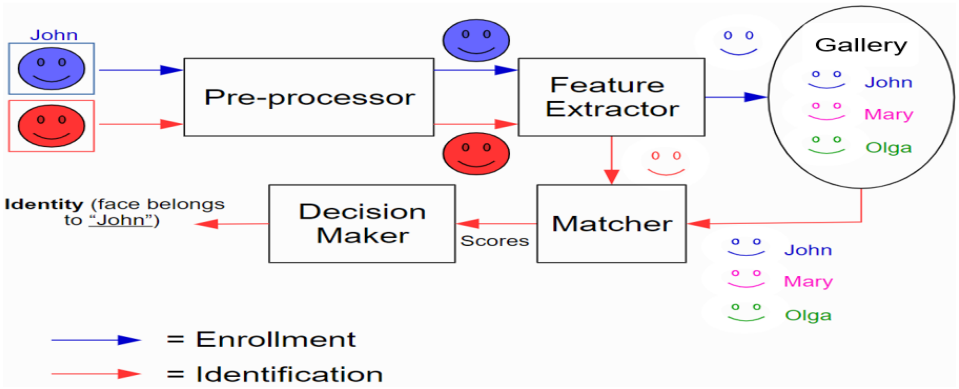



Figure 4.2: The Identification System[1]

## 4.1 Data Pre-Processing

As described on the website[1] the first step is the data pre-processor. Since the pictures were taken in an uncontrolled environment, there is a lot of unnecessary information such as the faceless background, which is an obstacle to face verification. To make the face verification task easier, all pictures are cleaned up in the pre-processing step so that noisy information is reduced. For example, the background noise is reduced by cropping it out. Additionally, but not required, all images can be rotated so that their faces are upright (Singh et al., 2012). In the pre-processing step we do an alignment on pictures from LFW. We get source coordinates from the package bob.db.lfw[3] and define two target coordinates by hand. The left eye has to be on the target coordinate (100, 134), and the right eye on the target coordinate (100, 90), where the first number is the y-coordinate and the second the x-coordinate. After that, the pictures go through an additional cropping so that they are 224x224 pixels in size.

[3] https://www.idiap.ch/software/bob/docs/bob/bob.db.lfw/v2.1.5/guide.html

## 4.2 Feature Extraction

After the pre-processing step, the images go through the feature extraction step.[1] Here, we extracted the facial attribute vectors for each picture from our AFFACT network. This facial attribute vector is an array and has a length of 40, in which all 40 attributes of an image are contained. These 40 attributes can be found in Figure 2.1.

## 4.3 Matching

The matching step is split into two parts: enrollment and scoring algorithms.[1]

### 4.3.1 Enrollment

As described on the website[1] and shown in Figure 4.3 before features of pictures are extracted or projected each of them has its corresponding ID because all pictures from the LFW database are labelled. Then the first half of the pictures from the test set in LFW are enrolled as a "model" in the gallery.

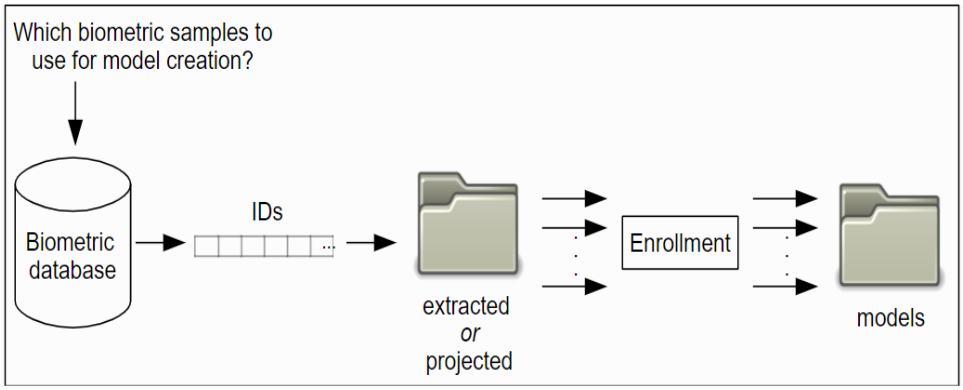



Figure 4.3: Enrollment[1]

### 4.3.2 Scoring

The goal of the step scoring is to have at the end a score value which shows how similar were two pictures to decide if they belong to the same person, as shown in Figure 4.4.[1] On one hand we use images from the training data set to calculate mean values and standard deviations for each attribute over several images, which are then needed when using images from the test data set to compute the score. This procedure is explained in Section 4.4. On other hand, we use them to train an SVM to compute the score of two images from the test data set. This procedure is explained in Section 4.5. Before describing how the score is calculated, it will be explained how the images from the training data set are arranged to use them.

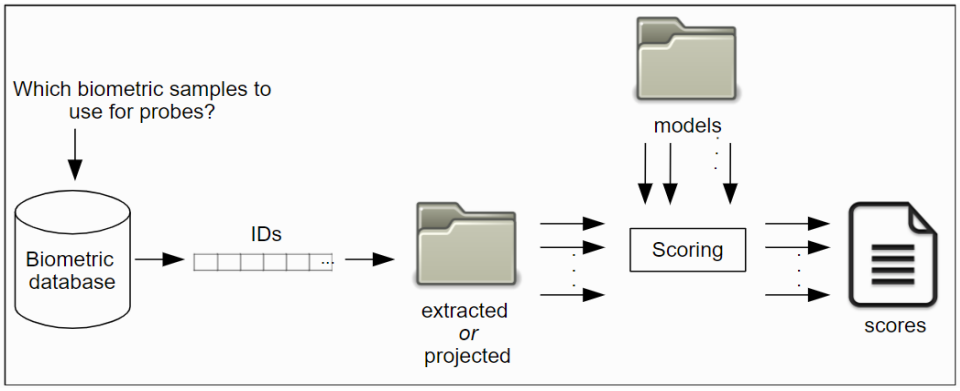



Figure 4.4: Scoring[1]

## Arrangement of the Images from the Training Data Set

Each image in the training data set has its own vector with a length of 40, in which all its 40 attributes are stored. This vector is extracted in such a way that it is available as an array with a length of 40. Based on the labels and the information from the matching data set, all arrays of images of one person are packed in a separate sub-array. Then all sub-arrays of all persons are combined in one main array called $training\_feature$ which is defined as:

$$training\_feature = \left[ \left[ \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix} \right], \left[ \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix}, \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix}, \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix} \right], \left[ \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix}, \begin{bmatrix} att_1 \\ att_2 \\ ... \\ att_{40} \end{bmatrix} \right] ... \right].$$

The first person, namely the first sub-array in the $training\_feature$ array, has only one image since it only has one array, the second person three images because it has three arrays, the third person two images because it has two arrays etc. Taking the $training\_feature$ array makes it easier for us to iteratively take two pictures of one person or of different people to compare their 40 facial attributes. How we take two images from the same person and from two different people is shown in Algorithm 1. We capture all possible pairs of images from the same person, but only a subset of all possible pairs of images from two different people, by creating a random index for the second image of the pair, as shown in Algorithm 1. When comparing two images we take their arrays and compare each of their face attributes: the first attribute of image 1 with the first attribute of image 2, etc. A calculation is made for each comparison and the results are packed into a new array. How these calculations are made is described in Sections 4.4 and 4.5.

## Score Calculation

The images from the test data set are used to calculate the score values. In order to achieve this, two images from it are iteratively picked out. The first one is the model and the second one is the probe image. Similar to the images from the training data set, each image in this set is also extracted as an array with a length of 40, in which all of its 40 attributes are stored. The difference is that while the labels from the training images are used to know their identity, the labels from the test images are not provided to the algorithm. The algorithm does not know whether the probe and model images are of the same person or from different people. Only the two arrays are needed and each of their face attributes is compared separately. The results of the calculations

**Algorithm 1:** Taking two images from same person or from two different people

```
for person_array in training_feature do
    if length(person_array) > 1 then
        for img1_array in person_array with index a do
            for img2_array in person_array with index b do
                if a < b then
                    img1_array and img2_array are from the same person
for person1_array in training_feature with index i do
    for img1_array in person1_array do
        r = random index from training_feature without index i;
        for person2_array in training_feature with index r do
            if i < r then
                for img2_array in person2_array do
                    img1_array and img2_array are from two different people
```

are summarized in a new array. The comparison of the two feature arrays provides a score value. How these calculations are made and how the score values are computed is described in Sections 4.4 and 4.5.

As described on the website[1] the score value shows how similar the model and probe images are, so that it can be deduced whether they are from the same person or not. Namely, a higher value of the score shows a larger similarity between the model and probe images and that they belong to the same person. A lower value of the score indicates less similarity and that they belong to different people. Since the images in the test data set are also labelled we are able to analyse how well the score was computed or how well an SVM was trained to classify two images as matched or mismatched pairs based on the score calculation.

## 4.4 Algorithms without Training an SVM

In this section we develop three algorithms to calculate the score values without using an SVM. In all three algorithm the negative sum from the new array is taken and returned as a score value. How the new array is calculated is described in the following subsections.

### Euclidean distance

In this algorithm, shown in Algorithm 2, the Euclidean distance between each attribute of the probe and model images is computed and shows how much each attribute differs between the model and probe images. If they are very different in both images, then their Euclidean distance will be very high, which then predicts that the two images differ greatly on that attribute. Otherwise, if the distance is very small, then the attribute values of both images are almost the same, which could be the case that the images come from the same person. The score value is then derived from the negative sum of all 40 distances between the 40 attributes of the model and probe images. The reason for taking the negative sum is that the larger the sum of all 40 distances is, the larger the difference between the probe and model images and the more negative it becomes. This means that the more negative the score value is, the more dissimilar the images are.

**Algorithm 2:** Euclidean_distance

Let p be the array with its 40 attributes of the probe and m of the model image ;
Let z be an empty array ;
**for** *i=1 till 40* **do**
    z[i] = $\sqrt{(p[i] - m[i])^2}$ ;
score = - sum(z);

## Distance Divided by Standard Deviation

In the previous algorithm, the distance between the attributes is taken from the two images. However, there are attributes that vary greatly for several images of one person and only calculating the distance is not very meaningful on its own. If two pictures of the same person are compared on an attribute which varies greatly and their distance is computed, then it can be that this computed value is still large and has an influence by summing up and makes the score value more negative. The standard deviation of this attribute is very high because it fluctuates extremely inside people. On other hand, there are attributes that do not vary greatly. If then two pictures of different people are compared on that attribute and their distance is computed, then it can be that this computed value is very small and has no influence by summing up all computed distances. Therefore, it does not make the score more negative even if the pictures are of different people. The standard deviation of that attribute is very low since it does not fluctuate extremely. For these reasons we have calculated an array $\sigma$ which contains all mean values over the standard deviations per person from the 40 attributes of training images, which is to be followed in the training part in Algorithm 3. For the score calculation we take the difference between each attribute of the two images individually, square that and divide it by the corresponding squared standard deviation in $\sigma$, which can be traced in the testing part in Algorithm 3.

**Algorithm 3:** Dist_div_by_std

# training part;
Let std_deviation be an empty array;
**for** *each person with at least two images in training_feature* **do**
    Let std_array be an empty array ;
    **for** *i=1 till 40* **do**
        std_array[i] = the standard deviation over all arrays of that person;
    append std_array in std_deviation array ;
$\sigma$ = the mean over all arrays in std_deviation;
# testing part;
Let p be the array with its 40 attributes of the probe and m of the model image ;
Let z be an empty array ;
**for** *i=1 till 40* **do**
    z[i] = $\frac{(p[i] - m[i])^2}{\sigma^2[i]}$;
score = - sum(z);

## Distance Divided by One of the Standard Deviations

Since there are attributes which vary greatly between images of the same person when they are present on an image but are very stable when they are absent or vice versa, we have divided

the standard deviations into two arrays in this algorithm. For example, the attribute "No Beard" is present in pictures of women and is very stable and hardly fluctuates. Whereas in pictures from men it is absent and fluctuates extremely because there are men who have no beard, a full beard or a half beard. Therefore, we made for each attribute a distinction. How we made this distinction can be traced in the training part in Algorithm 4. In $one\_std\_neg$ array there are standard deviations of attributes if the mean value of them is negative and in $one\_std\_pos$ if the mean value is positive. For the score calculation in the testing part these two arrays are needed. Instead of dividing the computed distance between the probe and model images by the same standard deviation as before, here the distance is divided by the standard deviation which is either the array $one\_std\_neg$ or $one\_std\_pos$. Namely, if the attribute of the probe image is negative, then the absolute difference between that attribute of the probe and model images is divided by the array $one\_std\_neg$ otherwise by the array $one\_std\_pos$. Here, too, the score value is the negative sum of all 40 calculated distances divided by the corresponding standard deviation. Compared to Algorithm 3 where for the score calculation the difference between the attribute of the probe and model images is squared and divided by the squared standard deviation, here the score is calculated by dividing the absolute difference between the attribute of the probe and model images by the standard deviation. The reason why we take the absolute difference between the attributes is that with the absolute difference the distance between them is calculated, which shows how strongly they differ from each other.

**Algorithm 4:** Dist_div_by_one_std

```
# training part;
for i=1 till 40 do
    Let negative_i and positive_i be empty arrays;
for each person with at least two images in training_feature do
    mean_array = the mean over all arrays;
    std_dev = the standard deviation over all arrays;
    for i=1 till 40 do
        if mean_array[i] < 0 then
            append std_dev[i] into the array negative_i;
        else
            append std_dev[i] into the array positive_i;
Let one_std_pos and one_std_neg be empty arrays;
for i=1 till 40 do
    mean_neg = mean of negative_i;
    append mean_neg into one_std_neg;
    mean_pos = mean of positive_i;
    append mean_pos into one_std_pos;
# testing part;
Let p be the array with its 40 attributes of the probe and m of the model image ;
Let z be an empty array ;
for i=1 till 40 do
    if p[i] < 0 then
        z[i] = |p[i]-m[i]| / one_std_neg[i] ;
    else
        z[i] = |p[i]-m[i]| / one_std_pos[i] ;
score = - sum(z);
```

## 4.5 Algorithms Training an SVM

Different than in the previous section, here we develop five algorithms that train an SVM.

As described on the following website[4] and by Singh et al. (2012) an SVM is very powerful because it is trained to learn from examples in order to make a binary classification and prediction. Moreover, it is memory efficient. Different kernel functions such as linear kernel, Radial Basis Function RBF kernel or polynomial kernel can be used.

In our thesis we use the RBF kernel. To train an SVM, two images of the same person or from different people are compared and a new array will be calculated. For each calculated array, a vector $\vec{t}$ is associated. If the new array was computed from arrays of the same person, its associated vector is $\vec{t} = (1)$. Otherwise, its associated vector is $\vec{t} = (-1)$. Then this new array and its associated vector $\vec{t}$ are put in the method $fit$ and train an SVM.[4] After training an SVM, a calculation is done between the probe and model images and is put into a new array. By adding the new array in the method $decision\_function$ the score value is computed.[4] If the score is below or equal to the threshold, the probe and model images are from the same person, and if it is above the threshold, they are from different people (Huang and Learned-Miller, 2014). How the new array is calculated is described in the following subsections.

### Absolute Distance and Multiplication

Similar to how Rudd et al. (2016) and Kumar et al. (2009) have compared two images with each other by first calculating the absolute difference between their attributes and also multiplying them and then packing the results into a vector, the new array between two images is calculated exactly the same. Namely the first forty entries of the array are the absolute difference between two arrays of two images and the other forty entries is the multiplication of them, which can be followed in the training part in Algorithm 5. After training an SVM the probe and model images are compared and also a new array will be calculated the same way between them, which can be traced in the testing part in Algorithm 5. Then this new array is put into the $decision\_function$ to compute the score value. The absolute difference between two arrays was carried out to calculate how much their attributes differ from each other. When multiplying two arrays, the first attributes are multiplied with each other, the second attributes with each other and so on. The result of the multiplication between strongly deviating values of the attributes of the two arrays such as values -2 and 2.5 is a very strong negative value and is therefore very low whereas the result of the multiplication between strongly matching values such as values 2.4 and 2.5 is a very high positive value. If the values of the attributes of the two arrays are between 0 and 1 or 0 and -1 such as 0.2 and 0.3, which do not make sufficient conclusion as to whether this attribute is present on the picture, and if they are multiplied together, the result is a smaller value in terms of amount. For these reasons, the newly calculated array with such values can be classified well.

### Absolute Distance and Multiplication Divided by Standard Deviation

This algorithm is similar to Algorithm 5 but with one difference. Because there are attributes that vary greatly or that are very stable for several images of one person we have calculated for the same reason as described for the Algorithm 3 the standard deviation for each attribute across the sub-arrays of the $F\_same$ array, which can be retraced in the training part in Algorithm 6. As already shown in Algorithm 5 one such sub-array contains the results of the absolute difference and the multiplication of two images of the same person. The calculated standard deviations are then put into the $std\_array$. After that, each sub-array in $F\_same$ and $F\_diff$ is divided by $std\_array$. In contrast to Algorithm 5, $F\_same$ and $F\_diff$ are only used to train an SVM

[4] https://scikit-learn.org/stable/modules/svm.html

**Algorithm 5:** Distance_multiplication

```
# training part;
Let F_same, F_diff, GT_same and GT_diff be empty arrays ;
for each person with at least two images in training_feature do
    for each two different images of that person do
        Let a and b be arrays of these two images ;
        s = |a − b|, m = a ⊙ b and t = (1) ;
        D = [s, m] ;
        append D into F_same and t into GT_same
for each two different people in training_feature do
    for each their images do
        Let a and b be arrays of these two images ;
        s = |a − b|, m = a ⊙ b and t = (-1) ;
        E = [s, m] ;
        append E into F_diff and t into GT_diff
F = [F_same, F_diff] and GT = [GT_same, GT_diff] ;
SVM.fit(F, GT) ;
# testing part;
Let p be the array with its 40 attributes of the probe and m of the model image ;
s = |p − m|, l = p ⊙ m;
F = [s, l] ;
SVM.decision_function(F)
```

after dividing their sub-arrays by the standard deviation. If the calculated value at one attribute between two images of the same person and the standard deviation of that attribute are large, then dividing the value by the standard deviation it gets smaller and thus clearly indicates that the two images from the same person do not differ much on that attribute. It is only the case that this attribute fluctuates very strongly. On other hand if the calculated value at one attribute between two images of different people and the standard deviation of that attribute are small, then dividing the value by the standard deviation it gets larger and thus clearly indicated that the two images from different people differ much on that attribute. It is only the case that this attribute is very stable. Thus, an SVM can be trained better and can make more meaningful predictions. The same applies for the score calculation, which can be traced in the testing part in Algorithm 6. After the absolute difference and multiplication between the probe and model images have been calculated and combined in an array, this array is also divided by the $std\_array$. Only after the division is the array put into the $decision\_function$.

## Absolute Distance and Multiplication Divided by One of the Standard Deviations

In this algorithm we summarize all the thoughts of Algorithms 4 and 6. Firstly, the absolute difference and a multiplication between two arrays of two images of the same person is calculated and the results are put in arrays called s and m, which can be followed in Algorithm 7. Then a standard deviation is calculated for each attribute over all arrays s and another standard deviation for each attribute over all arrays m. These standard deviations show how much attributes vary between images of the same person. These calculated standard deviations are then divided into two different arrays, depending on whether the corresponding mean value of its

**Algorithm 6:** Dist_mul_div_by_std

```
# training part;
Line 1-14 from Algorithm 5;
for i=1 till 40 do
    std_array[i] = standard deviation across all sub-arrays in F_same at index i;
Let new_F_same and new_F_diff be empty arrays ;
for each sub-array in F_same array do
    d = subarray / std_array ;
    append d in new_F_same
for each sub-array in F_diff array do
    d = subarray / std_array ;
    append d in new_F_diff
F = [new_F_same, new_F_diff] and GT = [GT_same, GT_diff] ;
SVM.fit(F, GT) ;
# testing part;
Let p be the array with its 40 attributes of the probe and m of the model image ;
s = |p − m|, l = p ⊙ m;
F = [s, l] ;
F_divided = F / std_array ;
SVM.decision_function(F_divided)
```

attribute is negative or positive. As already described for Algorithm 4 there are attributes that vary greatly when they are present on images but are very stable when they are absent or vice versa. Because of this, this separation is done so that the arrays s and m will later be divided by a more accurate standard deviation. At the end there are four arrays called $one\_std\_pos\_subtract$, $one\_std\_pos\_multiply$, $one\_std\_neg\_subtract$ and $one\_std\_neg\_multiply$ which are then used in the second part. Here the mean values for each attribute individually are checked for being negative or positive in order to divide the absolute difference and the multiplication between the two images by the corresponding standard deviation. For the score calculation, which is shown in Algorithm 8, the absolute difference and the multiplication between each attribute of the probe and model images is divided by the corresponding standard deviation based on whether that attribute of the probe is negative or positive.

## Chi-Square Distance and Multiplication Divided by One of the Standard Deviations

This algorithm is identical to Algorithms 7/8 but with one difference. While in Algorithms 7/8 the s array was calculated in such a way that the absolute difference between two arrays was taken, here the s array is the Chi-square distance between them. As shown in Algorithm 9, this calculation is done elementwise. The Chi-square distance is computed between two attributes to compare the two images on how similar they are on that attribute (Ren et al., 2019).

**Algorithm 7:** Dist_mul_div_by_one_std

Let std_pos_subtract, std_pos_multiply, std_neg_subtract and std_neg_multiply be empty arrays ;
Let subtract and multiply be empty arrays ;
**for** *each person with at least two images in training_feature* **do**
    mean_array = the mean over all arrays;
    **for** *each two different images of that person* **do**
        Let a and b be arrays of these two images ;
        $s = |a - b|$ and $m = a \odot b$ ;
        Append s in subtract and m in multiply;
    **for** *i=1 till 40* **do**
        std_subtract[i] = standard deviation over all subarrays[i] in subtract;
        std_multiply[i] = standard deviation over all subarrays[i] in multiply;
    **for** *i=1 till 40* **do**
        **if** *mean_array[i] < 0* **then**
            append std_subtract[i] into std_neg_subtract at index i;
            append std_multiply[i] into std_neg_multiply at index i;
        **else**
            append std_subtract[i] into std_pos_subtract at index i;
            append std_multiply[i] into std_pos_multiply at index i;
std_neg_subtract, std_neg_multiply, std_pos_subtract and std_pos_multiply are arrays with arrays;
Let one_std_pos_subtract, one_std_pos_multiply, one_std_neg_subtract and one_std_neg_multiply be empty arrays ;
**for** *each array in std_pos_subtract, std_pos_multiply, std_neg_subtract and std_neg_multiply respectively* **do**
    mean = mean of this array;
    append this mean into one_std_pos_subtract, one_std_pos_multiply, one_std_neg_subtract or one_std_neg_multiply respectively ;
# $2^{nd}$ part;
Let F_same and GT_same be empty arrays ;
**for** *each person with at least two images in training_feature* **do**
    mean_array = the mean over all arrays;
    **for** *each two different images of that person* **do**
        Let a and b be arrays of these two images ;
        $s = |a - b|$, $m = a \odot b$ and t = (1) ;
        Let subtract and multiply be empty arrays ;
        **for** *i=1 till 40* **do**
            **if** *mean_array[i] < 0* **then**
                subtract[i] = $\frac{s[i]}{one\_std\_neg\_subtract[i]}$ ;
                multiply[i] = $\frac{m[i]}{one\_std\_neg\_subtract[i]}$ ;
            **else**
                subtract[i] = $\frac{s[i]}{one\_std\_pos\_subtract[i]}$ ;
                multiply[i] = $\frac{m[i]}{one\_std\_pos\_subtract[i]}$ ;
        F = [subtract, multiply];
        append F into F_same and t in GT_same ;
Repeat the steps but with each two pictures of different people and the t = (-1) is appended into GT_diff and instead of naming the array F_same, name it F_diff ;
F = [F_same, F_diff] and GT = [GT_same, GT_diff] ;
SVM.fit(F, GT);

**Algorithm 8:** Dist_mul_div_by_one_std

Let p be the array with its 40 attributes of the probe and m of the model image ;
Let subtract and multiply be empty arrays ;
$s = |p - m|$, $l = p \odot m$;
**for** *i=1 till 40* **do**
    **if** $p[i] < 0$ **then**
        subtract[i] = $\frac{s[i]}{one\_std\_neg\_subtract[i]}$;
        multiply[i] = $\frac{l[i]}{one\_std\_neg\_multiply[i]}$ ;
    **else**
        subtract[i] = $\frac{s[i]}{one\_std\_pos\_subtract[i]}$;
        multiply[i] = $\frac{l[i]}{one\_std\_pos\_multiply[i]}$ ;
F = [subtract, multiply] ;
SVM.decision_function(F)

**Algorithm 9:** Chisquare_dist_mul

All lines from Algorithms 7 and 8;
Instead of $s = |a - b|$ or $s = |p - m|$ do this: ;
**for** *i=1 till 40* **do**
    s[i] = $\frac{1}{2}\frac{(a[i]-b[i])^2}{|a[i]|+|b[i]|}$ or s[i] = $\frac{1}{2}\frac{(p[i]-m[i])^2}{|p[i]|+|m[i]|}$

## Cosine Similarity and Multiplication Divided by One of the Standard Deviations

The last experiment is also nearly identical to Algorithms 7/8 but instead of calculating the absolute distance, the Cosine similarity between two arrays is calculated. This is shown in Algorithm 10 and is also done elementwise. With the Cosine similarity, as the name says, it is shown how similar two attributes of two images are. Each attribute from the two images is individually multiplied and then divided by the multiplication of the norm of the two arrays. If the attributes of the two images are very different and therefore very dissimilar, they are negatively correlated, and the cosine value is -1 (Jiawei et al., 2012). The closer the cosine value is to 1, the more similar the images are.

**Algorithm 10:** Cosine_sim_mul

All lines from Algorithms 7 and 8;
Instead of $s = |a - b|$ or $s = |p - m|$ do this: ;
**for** *i=1 till 40* **do**
    s[i] = $\frac{a[i] \cdot b[i]}{\|a\|\|b\|}$ or s[i] = $\frac{p[i] \cdot m[i]}{\|p\|\|m\|}$

# Chapter 5

# Experiments

After the score values between every pair of images in the test data set have been calculated, they are saved to a file called the score file. In each line of this file, the labels of the two images whose attributes were compared with each other and their calculated score value are listed. Based on the score value a decision is made as to whether the images are of the same person and created a match. How this decision is made is described in Section 5.1. Afterwards in Section 5.2 the results of the saved file of Algorithms 7/8 are shown and then in Section 5.3 the results of all algorithms are shown and analysed as curves and histograms.

## 5.1 Decision Making

As shown on the following website[1] the main difference between the verification and identification systems is the decision making step. In the verification system it decides whether the probe and model image match or not. This decision is made based on the score being larger or less than a defined threshold. If the score is above the threshold then it means that the probe and model images are classified as a matched pair, otherwise as a mismatched pair. The higher the score value, the more strongly the images are associated as images of a person and the lower the score value, the more strongly they are associated as images of different people. In other algorithms in which the score is the negative sum of all entries of the calculated array, the images are classified as matched pair if their score is not strongly negative. The more negative their score value, the more likely they are to be mismatched pairs.

As described on the website[1] the identification system has to find out which model image over all images in the gallery matches the best with the probe image and represents it preferably. And this is done by choosing the model with the highest similarity to the probe so that the score is the highest of all other score values that were calculated from other comparisons with other model images. Since a threshold is required to solve the face verification problem in order to make a decision based on it and the score value, solving the face verification problem is more difficult than solving the face identification problem (Seo and Milanfar, 2011). After a decision is made, the results are shown by using a performance metrics, which are shown in Figure 5.1. Some of these performance metrics are the False Match Rate (FMR), False Non Match Rate (FNMR), and Equal Error Rate (EER) for verification, and Identification Rate (IR) for identification. FNMR means that initially the probe and model images were coming from the same person but based on the score they are identified as different people. FMR is when the probe and model images were coming from different people but are identified as the same person. FNMR and FMR are

[1] https://www.idiap.ch/software/bob/docs/bob/docs/stable/bob/bob.bio.base/doc/

calculated by Derawi et al. (2011) as follows:

$$FNMR = \frac{\text{incorrectly rejected matched pairs}}{\text{total number of matched pairs}}$$

$$FMR = \frac{\text{incorrectly accepted mismatched pairs}}{\text{total number of mismatched pairs}}$$

If FNMR and FMR are equal, then the Equal Error Rate EER has been computed. This means the EER is defined so that FNMR = FMR.

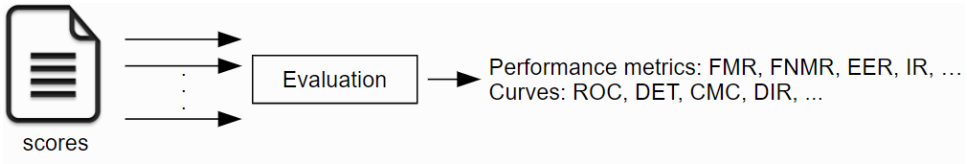



Figure 5.1: Performance Metrics[1]

At the very end the resulted performance can be shown in many representation curves like the Receiver Operating Characteristic (ROC), Detection Error Trade-off (DET) for verification and Cumulative Match Characteristics (CMC) for identification.[1] The x-axis of the ROC curve is the FMR and the y-axis is the 1-FNMR. For every calculated score value between two images in the test data set stored in the score file, the FMR and FNMR are recalculated and marked as threshold points on the coordinate system of the ROC curve. Based on the changing threshold and on the score value between two images, the images are redefined as matched or mismatched pairs. The ROC curve is drawn in such a way that it goes through all the thresholds, whereby the FMR and FNMR have a different value on each of them (Kumar et al., 2009). Therefore, ROC varies the threshold and plots the FNMR against the FMR. The ROC curve gives us the relative costs which have been made when the score says the opposite relation of two pictures as in the reality (Huang et al., 2008).

## 5.2 Results of the Score File

We have selected some score values of two images from the score file, which were calculated by Algorithms 7/8. For these algorithms we use the threshold of zero. In Figures 5.2 and 5.3 the images were taken from the person named Denzel Washington. When comparing the second picture with the fifth picture the resulting score is 1.697, but when comparing it with the fourth picture the resulting score is only 0.444. Comparing then this fourth picture of him with the picture of Sung Hong Choi the score is -1.160, shown in Figure 5.4. The score was predicted by the algorithms correctly because it is above the threshold when the images of the same person, namely of Denzel Washington, and it is below the threshold when images of different people, namely of him and of Sung Hong Choi, were compared.

When we compare pictures of the person named Abdullah Gul, shown in Figures 5.5 and 5.6, the score between the 13th and 14th pictures of him is 1.666 but between the 13th and 16th pictures it is -1.357. Since the second one is below the threshold, these two pictures were associated as mismatched pairs even they are from the same person. In Table 5.1 some extracted attributes of the 13th and 16th pictures of him are shown, which differ between these two images. Such attributes are "Big Nose", "Black Hair", "Bushy Eyebrows", "Chubby", "Double Chin", "Gray Hair",

"Mustache", "No Beard" and "Pointy Nose". While these attributes are defined as being present on one picture, they are absent on the other picture or vice versa. The reason for this is because the 16th picture of him includes his profile view, extraction of some attributes from the profile view is wrong and this leads to a wrong prediction. If this 16th picture of Abdullah Gul is compared with a picture of Steve Cox, shown in Figure 5.7, the score is -0.958, which is higher than the one in Figure 5.6. This is interpreted to mean that two images of the same person, namely the 13th and 16th pictures of Abdullah Gul, were more strongly associated as an unmatched pair than images of different people, namely the 16th picture of Abdullah Gul and the picture of Steve Cox.

| Nr | Attribute | 13th Picture | 16th Picture |
|---|---|---|---|
| 8 | Big Nose | 0.7831932 | -0.10086899 |
| 9 | Black Hair | -0.44110107 | 0.21394476 |
| 13 | Bushy Eyebrows | 0.4687491 | -1.1582938 |
| 14 | Chubby | 0.7989131 | -0.21953525 |
| 15 | Double Chin | 0.89657366 | -0.67920625 |
| 18 | Gray Hair | 0.6676535 | -0.594358 |
| 23 | Mustache | 0.9702361 | -0.20674163 |
| 25 | No Beard | -1.2136256 | 0.0773116 |
| 28 | Pointy Nose | -1.0214388 | 0.13217112 |

Table 5.1: Extracted Attributes of the 13th and 16th Pictures of Abdullah Gul

In Figures 5.8, 5.9 and 5.10 the opposite can be observed. Three pictures of the person named Annette Lu were taken, the first one is compared with the second one and with the third one. The score between the first and third images of Annette Lu is 0.380, shown in Figure 5.8, and the score between the first and second images of her is -0.058, shown in Figure 5.9. While in the first comparison her images are classified as matched pairs, in the second comparison they are classified as mismatched pairs. In Table 5.2 some extracted attributes, such as "Big Lips", "Double Chin", "High Cheekbones", "Narrow Eyes", "Pale Skin" etc., of the first and second pictures of her are shown, which also differ between these two images. Because the second picture of her includes her profile view, extraction of some attributes from the profile view is also wrong and leads to a wrong prediction. The difference in the "Pale Skin" attribute is due to the different lighting conditions on both images. It is interesting that based on the extracted "Wearing Hat" attribute, Annette Lu is wearing a hat on the first picture, but this is not true. One possible reason for this could be the poor lighting conditions in the first picture. If then her first image is compared with the image of Svetlana Belousova, shown in Figure 5.10, the score is above the threshold, because it is 0.296. While two pictures of the same person, namely from Annette Lu in Figure 5.9, were classified as unmatched pair, one picture of this person and another picture of another person were classified as matched pair.

Another example where images of different people are classified as images of the same person is shown in Figure 5.11. One image is from Dennis Kozlowski and the other is from James Ivory. Their score is 1.243 and therefore above the threshold. The reason for this could be that both have light skin, a half-bald to a complete bald head and narrow lips. Examples where two faces are from different people but have similar attributes show that this is a limitation of face recognition with facial attributes. Any algorithm based on the comparison of attributes must fail and classify them as the same person.

These examples show that Algorithms 7/8 sometimes make wrong predictions. In some cases, as in Figures 5.6 and 5.9, two images of the same person were classified as unmatched, and in other cases, as in Figures 5.10 and 5.11, two images of different people were classified as matched

| Nr | Attribute | 1st Picture | 2nd Picture |
|---|---|---|---|
| 7 | Big Lips | 0.12129111 | -1.4537106 |
| 8 | Big Nose | 0.5174873 | -0.7740385 |
| 15 | Double Chin | 0.76103044 | -0.32412586 |
| 20 | High Cheekbones | 0.3265112 | -0.35287246 |
| 24 | Narrow Eyes | 0.50965536 | -0.049525328 |
| 27 | Pale Skin | 0.40810573 | -0.82514215 |
| 32 | Smiling | 0.8515469 | -0.8569647 |
| 34 | Wavy Hairs | -0.21393725 | 0.5017383 |
| 36 | Wearing Hat | 0.23983678 | -1.0175481 |

Table 5.2: Extracted Attributes of the 1st and 2nd Pictures of Annette Lu

pairs due to the variation in the uncontrolled environment in which the images were captured. One possible reason why two images of different people could be classified as matched pairs is that as mentioned in Section 4.3.2 and shown in Algorithm 1 we capture all possible pairs of images from the same person, but only a subset of all possible pairs of images from two different people. Thus, we have trained an SVM in the algorithms with more comparisons of images of the same person than of different people. To be exact, we have taken 217'637 pairs of images from the same person and 23'255 pairs of images from different people. Therefore, an SVM might not be trained enough to identify images as mismatched pairs as it had less input from comparisons between them.

(a) Denzel Washington's 2nd Picture

(b) Denzel Washington's 5th Picture

Figure 5.2: Denzel Washington's 2nd and 5th Pictures With Score 1.697

(a) Denzel Washington's 2nd Picture

(b) Denzel Washington's 4th Picture

Figure 5.3: Denzel Washington's 2nd and 4th Pictures With Score 0.444

(a) Denzel Washington's 4th Picture

(b) Sung Hong Choi's Picture

Figure 5.4: Denzel Washington's 4th and Sung Hong Choi's Pictures With Score -1.160

(a) Abdullah Gul's 13th Picture

(b) Abdullah Gul's 14th Picture

Figure 5.5: Abdullah Gul's 13th and 14th Pictures With Score 1.666

(a) Abdullah Gul's 13th Picture

(b) Abdullah Gul's 16th Picture

Figure 5.6: Abdullah Gul's 13th and 16th Pictures With Score -1.357

(a) Abdullah Gul's 16th Picture

(b) Steve Cox's Picture

Figure 5.7: Abdullah Gul's 16th and Steve Cox's Pictures With Score -0.958

(a) Annette Lu's 1st Picture

(b) Annette Lu's 3rd Picture

Figure 5.8: Annette Lu's 1st and 3rd Pictures With Score 0.380

(a) Annette Lu's 1st Picture

(b) Annette Lu's 2nd Picture

Figure 5.9: Annette Lu's 1st and 2nd Pictures With Score -0.058

(a) Annette Lu's 1st Picture

(b) Svetlana Belousova's Picture

Figure 5.10: Annette Lu's 1st and Svetlana Belousova's Pictures With Score 0.296

(a) Dennis Kozlowski's Picture

(b) James Ivory's Picture

Figure 5.11: Dennis Kozlowski's and James Ivory's Pictures With Score 1.243

## 5.3 Results of All Algorithms as ROC Curves and Histograms

After the score has been calculated for each pair of images and the decision has been made as to whether they should be classified as matched or mismatched pairs we use the Receiver Operating Characteristic (ROC) and histograms as representation curves to show the results.

### 5.3.1 ROC Curves

The ROC curve shows how well the algorithm was developed and how well it computed the score between the model and probe images from the test data set. In algorithms described in Section 4.5, in which an SVM was trained with the training data set, the ROC curve shows additionally how well an SVM was trained. The goal of the algorithms is that after the calculation of the score values, the values of FNMR and FMR are as small as possible. Because the x-axis of the curve is the FMR values and the y-axis is the 1-FNMR values, the values on the x-axis must be as small as possible and on the y-axis as large as possible. The curve that lies above the other curve(s) is the better one, because with the same value on the x-axis the value on the y-axis is higher and thus the FNMR value is lower. The highest value on the x-axis is when the FMR value is one and on the y-axis when the FNMR value is zero. This is the case when the lowest score from mismatched pairs from the score file has been selected as the threshold. Thus, all other score values are above the threshold and all images have been classified as matched pairs even if some are of different people. In some very rare circumstances where the lowest score from matched pairs is lower than the lowest score from mismatched pairs, the 1-FNMR value is below one and thus FNMR value is above zero. The lowest value on the x-axis and on the y-axis is the opposite. Here the highest score from mismatched pairs from the score file is the threshold and therefore all other score values are below the threshold so that all images have been classified as mismatched pairs even if some are from the same person. It happens quite regularly that the scores from matched pairs are higher than the highest score from mismatched pairs and, thus, the 1-FNMR value does not decrease to zero. The ROC curves of all algorithms are shown in Figure 5.12.

In Figure 5.12(a) the ROC curves of the three algorithms, in which no SVM has been trained, namely Algorithms 2, 3 and 4, are drawn. Algorithm 2 is the "Euclidean_distance", Algorithm 3 is the "Dist_div_by_std" and Algorithm 4 is "Dist_div_by_one_std" curve. As you can see, all three curves look very much the same at the beginning and at the end of them. They only differ minimally in the middle of the curves. The green curve "Dist_div_by_std" lies above the other curves and the blue curve "Euclidean_distance" lies below the other curves. This means, given the same value on the x-axis, the value on the y-axis of the green curve is higher and thus the FNMR value is lower. With the blue curve it is the other way round: the value on the y-axis is lower and thus the FNMR value is higher. From this it can be ruled out that Algorithm 4 can predict images as matched and mismatched pairs better than Algorithms 2 and 3. The reason for this may be that in Algorithm 4 each calculated distance between two attribute values of two images was divided by the corresponding standard deviation of that attribute. In Algorithm 3 the distance is also divided by the standard deviation. However, the difference is that the distances are divided by the same standard deviation of that attribute. Whereas in Algorithm 4 the distance is divided by one of two standard deviation, depending on whether the mean of the attribute is negative or positive. This separation into two separate standard deviations was made because some attributes vary greatly between images of the same person and have a very large standard deviation if they are present, or they are very stable and have a small standard deviation if not, or the other way around. With this additional separation of the standard deviations in Algorithm 4 it is clearer whether two images are strongly or weakly similar based on that attribute: if an attribute

varies greatly and the difference between images of the same person is very large, then when that difference is divided by the standard deviation, the difference becomes smaller and better assertions are made that the images are from the same person. The opposite is also true. In Table 5.3 the standard deviations from Algorithm 3 and 4 are shown: in rows with caption "Negative Std" are the standard deviations of attributes listed which are absent on images, and in rows with caption "Positive Std" are the standard deviations of attributes listed which are present on images. Both are made in Algorithm 4. In rows with caption "Normal Std" are the standard deviations listed which are made in Algorithm 3. If attributes listed in Table 5.3 are absent, then they have a much lower standard deviation than if they are present. This shows that these attributes are very stable if they are absent and vary greatly if not. Whereas the standard deviations made in Algorithm 3 only says how much the attributes vary for several images of one person, regardless of whether the attributes are present on images or not. Therefore Algorithm 4 can divide the distance by a more precise standard deviation and predict the matched and mismatched pairs better than the other two algorithms and its curve lies above the other two curves.

| Nr | Attribute | Negative Std | Positive Std | Normal Std |
|---|---|---|---|---|
| 3 | Attractive | 0.11748319 | 0.3013416 | 0.12648979 |
| 6 | Bangs | 0.104566604 | 0.3276349 | 0.11775473 |
| 19 | Heavy Makeup | 0.08733079 | 0.4011784 | 0.11516348 |
| 30 | Rosy Cheeks | 0.12599859 | 0.40829816 | 0.12742904 |
| 35 | Wearing Earrings | 0.12779066 | 0.34810236 | 0.15402709 |
| 36 | Wearing Hat | 0.147307 | 0.5428519 | 0.17470121 |
| 37 | Wearing Lipstick | 0.07497735 | 0.34208924 | 0.11152476 |
| 38 | Wearing Necklace | 0.13913506 | 0.32554114 | 0.15613836 |

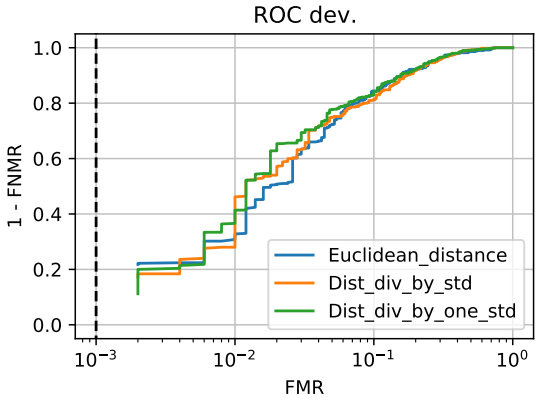
Table 5.3: The Standard Deviations calculated in Algorithm 3 and 4

The ROC curves of algorithms, in which an SVM has been trained, are shown in Figures 5.12(b) and 5.12(c). In the first one the curve of Algorithm 5 is the "Distance_multiplication", of Algorithm 6 the "Dist_mul_div_by_std" and of Algorithms 7/8 the "Dist_mul_div_by_one_std". As already mentioned in Section 4.5 Algorithm 5 is the baseline algorithm from Kumar et al. (2009) and Rudd et al. (2016). In the second one the curve of Algorithms 7/8 is also the "Dist_mul_div_by_one_std", of Algorithm 9 the "Chisquare_dist_mul" and of Algorithm 10 the "Cosine_dist_mul".

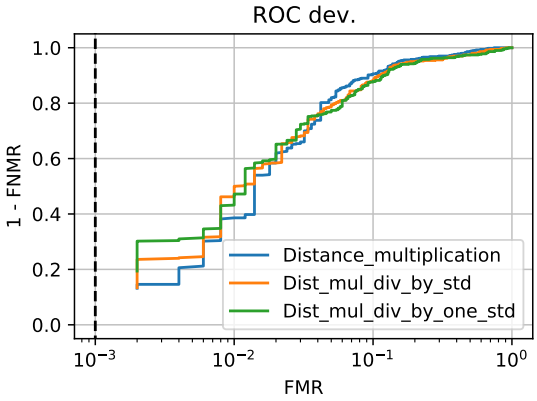
At the beginning of the curves in Figure 5.12(b), when the values on the x-axis and y-axis are low, the green curve "Dist_mul_div_by_one_std" lies above the others curves and the blue curve "Distance_multiplication" below. This shows that given the same value on the x-axis, the value on the y-axis is higher for the green curve and lower for the blue one. The higher the value on the y-axis, the lower the value of the FNMR. That means that the curve of Algorithms 7/8 cuts better than the curves of Algorithm 5, which is the baseline algorithm from Kumar et al. (2009) and Rudd et al. (2016), and Algorithm 6. The reason for this is the same as before, that in this algorithm the calculated distance is divided by one of two standard deviation. Towards the middle of the curves the three curves are almost the same and a difference can hardly be seen. Whereas towards the end of the curves, when the x-axis and y-axis values are high, the blue curve lies above the others and then all three curves are again almost similar and no difference can be seen.

The difference of the three algorithms in Figure 5.12(c) is that an absolute distance between two attribute arrays of two images is calculated in algorithm of curve "Dist_mul_div_by_one_std" whereas in algorithm of curve "Chisquare_dist_mul" the Chi-square distance and of curve "Cosine_sim_mul" the Cosine similarity. The algorithm of curve "Cosine_sim_mul" lies below all other curves and therefore predicts the score values the worst. Thus, the calculation of the Cosine

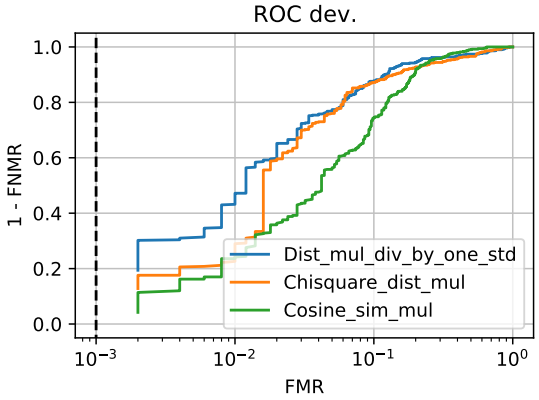
similarity between the attribute arrays can not infer very much from whether two images belong to the same person. At the beginning of the curves, where the values of the axes are small, the curve "Dist_mul_div_by_one_std" is above the others and towards the end of the curves, when the values of the axes increase, the curves "Dist_mul_div_by_one_std" and "Chisquare_dist_mul" hardly differ. With a very small difference to curve of Algorithm 9, the curve of Algorithms 7/8 lies above all other curves and therefore Algorithms 7/8 best predict the score values between the model and probe images. Thus, computing the absolute difference between two attribute arrays from two images instead of the Chi-square distance or Cosine similarity trains an SVM better so that better predictions can be made to classify two images from the test data set into matched or mismatched pairs.




(a) Algorithms Without Training an Svm




(b) Algorithms Training an SVM




(c) Algorithms Training an SVM

Figure 5.12: ROC Curves From All Algorithms

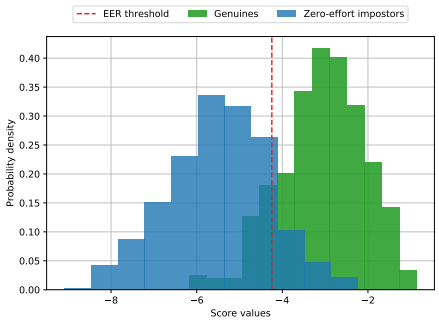
### 5.3.2 Histograms

In the histogram representation the score values are shown on the x-axis and the probability density of the matched or mismatched pairs for each score value on the y-axis. The matched pairs are called genuines and are shown in colour green, and the mismatched pairs are called zero-effort impostors and are shown in colour blue. The overlap of them can be seen in colour

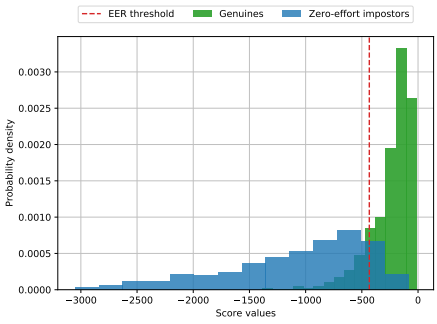
turquoise. The EER threshold is shown as a dashed red line. At this EER threshold the FNMR and FMR are equal. In the turquoise area to the right of the EER threshold, the FMR value is higher than the FNMR value and on the left the opposite. Since either the FMR or FNMR value is higher in these overlaps, these overlaps show how well the algorithms have classified the pairs. The larger the overlap, the higher the value of the errors and the worse the algorithm makes the predictions.

In Figure 5.13 the histograms of the three algorithms, in which no SVM has been trained, are shown. Since in these algorithms the score is the negative sum of all entries in the calculated array, the score values on the x-axis in all three histograms are negative. The less negative the score value, the more likely the images are classified as matched pairs and the more negative, the more likely as mismatched pairs. In histogram of Algorithm 2 shown in Figure 5.13(a) the range of the score values is between -9 and 0, of Algorithm 3 shown in Figure 5.13(b) between -3000 and 0 and of Algorithm 4 shown in Figure 5.13(c) between -250 and 0. Since the two axes of the three algorithms differ greatly from each other, it is difficult to rule out in which histogram the overlap of genuines and zero-effort impostors is the smallest. Therefore, we can not conclude from the histograms alone which algorithm has classified the pairs best.

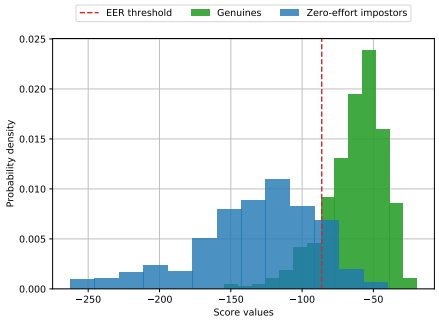
All other algorithms, in which an SVM has been trained, are shown in Figure 5.14. In all histograms the EER threshold is negative and lies between -1 and 0. The threshold value of zero, which we define for our trained SVM, does not match the EER threshold value. The range on the x-axis is between -4 and 3, except in Figure 5.14(e) of Algorithm 10 it is between -6 and 6. While the x-axis is similar for most of the histograms, the y-axis is different. In these histograms in Figure 5.14, too, it is difficult to select the histogram with the least amount of overlap in order to consider its algorithm to be the best.




(a) Euclidean_distance (Algorithm 2)

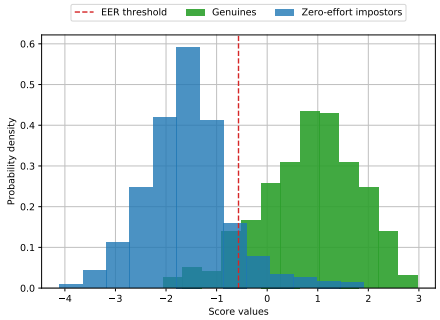



(b) Dist_div_by_std (Algorithm 3)

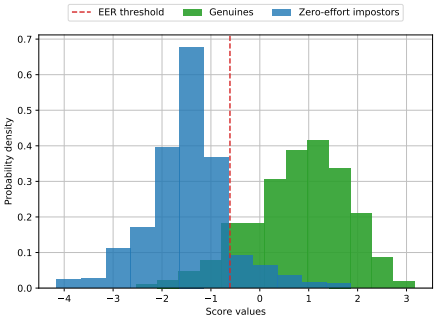



(c) Dist_div_by_one_std (Algorithm 4)

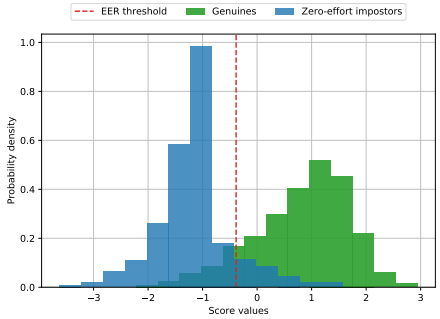
Figure 5.13: Histograms From Algorithms without Training an SVM



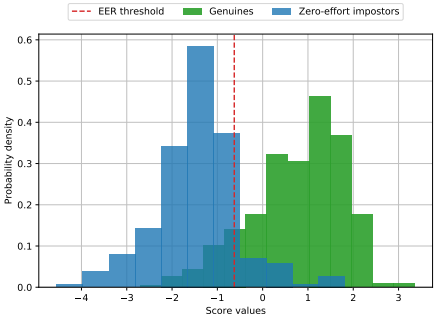

(a) Distance_multiplication (Algorithm 5)



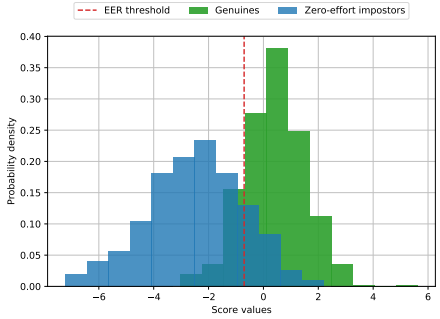

(b) Dist_mul_div_by_std (Algorithm 6)




(c) Dist_mul_div_by_one_std (Algorithms 7/8)




(d) Chisquare_dist_mul (Algorithm 9)




(e) Cosine_sim_mul (Algorithm 10)

Figure 5.14: Histograms From Algorithms Training an SVM

# Chapter 6

# Limitations

Even if the LFW database contains a large amount of pictures with different people and thus gives us the opportunity to compare the attributes between the same people and between different people, the use of these pictures still has limitations.

As discussed by Ohlyan et al. (2013) since the pictures were taken in a natural environment they vary in many external factors. One such factor is the pose angle of the faces. As seen in Figure 5.6 one picture shows the frontal view and the other shows the profile view of the same person and they were classified as mismatched pairs even though they are of the same person. Therefore, pose variation can lead to faulty face verification. Another factor is the changing in light intensity, which can affect the attributes of two pictures of the same person. Ohlyan et al. (2013) have shown that the difference between the attributes of two images of the same person, which differ greatly in terms of lighting conditions, is larger than the difference between the attributes of images from different people with no light variation. In addition, the images can differ in image quality. This is caused by the use of different cameras or if the face region on a picture is very small and by cropping and re-scaling it, the quality of the picture decreases (Ohlyan et al., 2013; Ling et al., 2010). Poor quality or different illumination of the images makes the comparison between the attributes of two images difficult (Ling et al., 2010).

Due to these variations, caused by external factors, between the images individual attributes can not be correctly detected on the images. If faces can only be seen from the profile view, then it is difficult to perfectly detect all attributes on that image. An example of this are the pictures of Abdullah Gul in Figure 5.6. His extracted attributes are shown in Table 5.1. While the "Big Nose", "Bushy Eyebrows", "Chubby", "Double Chin", "Gray Hair" and "Mustache" attributes are extracted as being present on the 13th picture, they are extracted as being absent on the 16th picture. Whereas the "Black Hair", "No Beard" and "Pointy Nose" attributes are extracted as being absent on the 13th picture, they are extracted as being present on the 16th picture. If you compare the two images, you can see that the attributes were extracted correctly on the 13th image and incorrectly on the 16th image. Since on the 16th image the profile view of him was taken, it limits the prediction of the attributes. In addition, it is possible that some attributes are extracted in such a way that they are considered to be present on the images even though they are not at all. For example, in Figure 5.9(a) Annette Lu is recognized as if she is wearing a hat even though she is not wearing one. The reason could be the poor lighting conditions. Thus, feature extraction can be influenced by the pose-angle variation, illumination changes or background noises (Soni et al., 2013). Because of these variations it is difficult to extract the attributes correctly (Kumar et al., 2009).

Another limitation of face recognition is that two faces from different people can have similar attributes, such as faces of images shown in Figure 5.11. As mentioned in Section 5.2 any algorithm must fail and classify them as the same person. Therefore, it is sometimes very difficult to correctly solve the face verification problem when comparing the two images to decide whether

they belong to the same person.

# Chapter 7

# Future Work

Future work should definitively focus primarily on reducing the discussed limitations in Section 6 as much as possible. Variation in illumination and in pose angles can have a large influence on the attributes of the images. There is a data set called Multi-PIE that was defined by Günther et al. (2016). They have run experiments on this data set to check which are the most important effects on face recognition to group images that differ only in one effect. Similar to Günther et al. (2016) one possibility would be to group pictures that only differ in their lighting conditions in one set and pictures that only differ in pose angles in another set. The only difference is using the facial attributes from the images in the LFW database, as Günther et al. (2016) have not used facial attributes in their experiments. Images in the two sets are then divided into matched pairs and mismatched pairs according to the LFW database structure. Since images from the same person differ either in light intensity or in the pose angle, an SVM can better learn how and which attributes vary between two images and which attributes do not change significantly and are unique to that person. Therefore, keeping the illumination and pose angle variations apart the challenge caused by them can be resolved and make face verification easier (Sao and Yegnanarayana, 2007).

Due to time constraints, we could not run the algorithms in View 2 which measures the performance of the algorithm ran in View 1. The images will be separated into ten subsets, nine of them are used as training and one as test data set. The algorithm will be run ten times and a mean accuracy and standard error of the mean are calculated for each one, as described in Section 3.2. These calculations show which subset and which algorithm have the smallest standard error of the mean, and therefore best calculate the score value of two images.

Another future work would be to consider age variations in images from LFW, as people in these images are of different ages and facial attributes change with age (Ling et al., 2010). People on pictures that were defined as matched pairs in LFW have the same gender and ethnicity and their age difference could be small. Whereas people on pictures that were defined as mismatched pair differ in gender and ethnicity and their age difference could be large.

According to authors Zheng et al. (2017) to take the age variations into account, the Cross-Age LFW (CALFW)[1] database can be used instead of the LFW database. The CALFW database is the same as the LFW database but has an additional step that measures the age for each image. Then pairs of the same person who have a large age difference and pairs of different people who are of the same race and gender are selected. Figure 7.1(a) shows pairs of the same person from LFW and Figure 7.1(b) from CALFW. Comparing the two figures shows that pairs in CALFW have a larger age difference than pairs in LFW. A larger age difference shows many attribute differences between two images of the same person at the same time and therefore an SVM can be better trained by learning all possible differences between attributes. Figure 7.2(a) shows pairs of different people from LFW and Figure 7.2(b) from CALFW. Comparing these two figures shows

[1] http://www.whdeng.cn/CALFW/CALFW.html

that pairs in CALFW have same gender and race than pairs in LFW, which reduces the influence of attribute difference between pairs from different people in face verification. Thus, attributes that differ between two images of different people, regardless of gender and race, become more significant and lead to better exclusion.


(a) Pairs of the Same Person from LFW


(b) Pairs of the Same Person from CALFW

Figure 7.1: The Comparison of Pairs of the Same Person from LFW and CALFW (Zheng et al., 2017)

(a) Pairs form Different People from LFW

(b) Pairs from Different People from CALFW

Figure 7.2: The Comparison of Pairs from Different People from LFW and CALFW (Zheng et al., 2017)

# Chapter 8

# Conclusions

Using the images from LFW gives us the opportunity to have many images of many different people of different ethnicity, age, gender, race, etc. For each image, we have extracted their 40 facial attributes from our AFFACT network. To solve the face verification problem of whether two images belong to the same person, we have compared each attribute individually between the two images. The comparison was carried out in such a way that distance-based calculations such as the Euclidean distance, the absolute distance, the multiplication, the Chi-square distance or the Cosine similarity were made between the attributes. With the images from the training data set, we calculated mean values and standard deviations for each attribute using images of the same person or from different people, and we also trained an SVM. Using the images from the test data set, we then calculated the score values and based on these, the images were classified as matched or mismatched pairs. To show how well the images were classified, we showed the results as ROC curves and histograms. After plotting the ROC curves for each of our developed algorithms, no significant difference could be found between them, except that the algorithm that calculated the absolute distance instead of the Chi-square distance or the Cosine similarity could make significantly better predictions about the classification of the images. With very little difference to the other curves, the algorithm, in which the calculations were divided by one of the two calculated standard deviations, performed better. Because there are attributes which vary greatly between images of the same person when they are present on images but are very stable when they are absent or the other way around, the standard deviations of these attributes were split up. Thus, the calculations can be divided by a more precise standard deviation. As can be seen from the curves in Figure 5.12, the curves are very close together and it is not possible to draw any conclusions as to which algorithm can by far make the best prediction.

We also noticed that some pictures were classified as matched pairs although they come from different people and others as mismatched pairs even though they come from the same person. As a result, algorithms sometimes could not classify them properly and made wrong predictions. The reason is that the images from the LFW database were taken in an uncontrolled environment and thus show natural variations in pose, lighting, expression, focus, resolution and camera quality. These variations cause that attributes can not be extracted correctly. In addition to the reason of the variation, which can limit the prediction, there were images with very similar attributes, which were classified as matched pairs even though they came from different people.

In summary, it can be said that when using the images from LFW, some limitations must be considered, which restrict the correct extraction of the attributes and the comparison between two images. If these limitations could be reduced, the face verification problem could be better solved in the future and correct predictions could be made to correctly classify two images.

# List of Figures

# List of Tables

# Bibliography


Berg, T. L., Berg, A. C., Edwards, J., Maire, M., White, R., Teh, Y.-W., Learned-Miller, E., and Forsyth, D. A. (2004). Names and faces in the news. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699.

Derawi, M. O., Yang, B., and Busch, C. (2011). Fingerprint recognition with embedded cameras on mobile phones. In *International conference on security and privacy in mobile information and communication systems*, pages 136–147. Springer.

Günther, M., Rozsa, A., and Boult, T. E. (2017). AFFACT: Alignment-free facial attribute classification technique. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 90–99. IEEE.

Günther, M., Shafey, L. E., and Marcel, S. (2016). Face recognition in challenging environments: An experimental and reproducible research survey. In *Face Recognition Across the Imaging Spectrum*, pages 247–280. Springer.

Guo, G. and Zhang, N. (2019). A survey on deep learning based face recognition. *Computer Vision and Image Understanding*, 189:102805.

Huang, G., Mattar, M., Lee, H., and Learned-Miller, E. G. (2012). Learning to align from scratch. In *Advances in Neural Information Processing Systems*, pages 764–772.

Huang, G. B., Jain, V., and Learned-Miller, E. (2007). Unsupervised joint alignment of complex images. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Huang, G. B. and Learned-Miller, E. (2014). Labeled faces in the wild: Updates and new reporting procedures. *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, 14(003).

Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, Marseille, France. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie.

Jiawei, H., Micheline, K., and Jian, P. (2012). 2 - getting to know your data. In *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 39–82. Morgan Kaufmann, Boston, third edition edition.

Kortli, Y., Jridi, M., Al Falou, A., and Atri, M. (2020). Face recognition systems: A survey. *Sensors*, 20(2).

Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *2009 IEEE 12th international Conference on Computer Vision*, pages 365–372. IEEE.

Ling, H., Soatto, S., Ramanathan, N., and Jacobs, D. W. (2010). Face verification across age progression using discriminative methods. *IEEE Transactions on Information Forensics and security*, 5(1):82–91.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738.

Ohlyan, S., Sangwan, S., and Ahuja, T. (2013). A survey on various problems & challenges in face recognition. *International Journal of Engineering Research & Technology (IJERT)*, 2(6):2533–2538.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In Xie, X., Jones, M. W., and Tam, G. K. L., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12. British Machine Vision Association.

Ren, H., Xiao, S., and Zhou, H. (2019). *A Chi-square distance-based Similarity Measure of single-valued Neutrosophic Set and Applications*. Infinite Study.

Rudd, E. M., Günther, M., and Boult, T. E. (2016). MOON: A mixed objective optimization network for the recognition of facial attributes. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *European Conference on Computer Vision*, pages 19–35, Cham. Springer.

Ruiz-del Solar, J., Verschae, R., and Correa, M. (2009). Recognition of faces in unconstrained environments: A comparative study. *EURASIP Journal on Advances in Signal Processing*, 2009:1–19.

Sao, A. K. and Yegnanarayana, B. (2007). Face verification using template matching. *IEEE Transactions on Information Forensics and Security*, 2(3):636–641.

Seo, H. J. and Milanfar, P. (2011). Face verification using the LARK representation. *IEEE Transactions on Information Forensics and Security*, 6(4):1275–1286.

Singh, A., Singh, S. K., and Tiwari, S. (2012). Comparison of face recognition algorithms on dummy faces. *The International Journal of Multimedia & Its Applications*, 4(4):121.

Soni, N., Kumar, M., and Mathur, G. (2013). Face recognition using SOM neural network with different facial feature extraction techniques. *International Journal of Computer Applications*, 76(3):7–11.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *European Conference on Computer Vision*, pages 499–515, Cham. Springer.

Zhang, N., Paluri, M., Ranzato, M., Darrell, T., and Bourdev, L. (2014). PANDA: Pose aligned networks for deep attribute modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1644.

Zheng, T., Deng, W., and Hu, J. (2017). Cross-Age LFW: A database for studying cross-age face recognition in unconstrained environments. *CoRR*, abs/1708.08197.