BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

RESEARCH IN THE FIELD OF BIOMETRIC DETECTION AND RECOGNITION OF INDIVIDUALS USING FACIAL IMAGE DATA

VÝZKUM V OBLASTI BIOMETRICKÉ DETEKCE A ROZPOZNÁVÁNÍ JEDNOTLIVCŮ POMOCÍ OBRAZOVÝCH DAT OBLIČEJE

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **Ing. TOMÁŠ GOLDMANN**
AUTOR PRÁCE

SUPERVISOR — **Prof. MARTIN DRAHANSKÝ**
ŠKOLITEL

BRNO 2023

Abstract

Biometric recognition has long since become a common concern in various fields of study, including forensics, anthropometry, biometrics, and computer science. This thesis focuses on the development of an approach to create datasets for the evaluation of face recognition algorithms, with an emphasis on the preservation of facial features. Such datasets open up new possibilities for the evaluation of face recognition algorithms, which were previously hindered by the limited sample size of the datasets usually used. Through extensive research in the field of face recognition algorithms and modern neural network techniques, algorithms for face detection and recognition on embedded devices have been developed. These algorithms are based on the EfficientNet feature extractor.

Abstrakt

Biometrické rozpoznávání osob se stalo běžnou součástí několika vědních oborů, včetně kriminalistiky, antropometrie, biometrie a informatiky. Práce se zaměřuje na vytvoření inovativního přístupu generování datových sad pro evaluaci algoritmů pro rozpoznávání osob podle obličeje. Na rozdíl od dosavadních přístupů naše řešení zachovává obličejové rysy. Vygenerované datové sady přinášejí nové možnosti pro hodnocení algoritmů rozpoznávání podle obličeje, které je jinak těžce realizovatelné z důvodu malého množství dat. Dále v této práci řešíme akceleraci algoritmů pro rozpoznávání osob na základě obličejových dat pomocí vestavěných zařízení. Především jsme se zaměřili na zhodnocení možností, které přináší neuronová síť pro extrakci příznaků EfficienNet.

Keywords

face, face detection, face recognition, neural network, Neural Processing Unit, generovaný dataset, EfficientNet, RetinaFace, ArcFace, MagFace

Klíčová slova

obličej, detekce obličeje, rozpoznávání podle obličeje, neuronové sítě, Neural Processing Unit, generated dataset, EfficientNet, RetinaFace, ArcFace, MagFace

Reference

GOLDMANN, Tomáš. *Research in the field of biometric detection and recognition of individuals using facial image data.* Brno, 2023. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Prof. Martin Drahanský

Research in the field of biometric detection and recognition of individuals using facial image data

Declaration

I hereby declare that this dissertation thesis was prepared as an original work by the author under the supervision Professor Martin Drahanský. The supplementary information was provided by Associate Professor Petra Urbanová. I have cited all literary sources, publications, and other resources from which I drew information.

.....
Tomáš Goldmann
September 12, 2023

Acknowledgements

I would like to express my sincere gratitude to the many individuals and organizations that played a critical role in the completion of this dissertation.

First and foremost, I am deeply indebted to my supervisor, Profesor Martin Drahanský, whose expertise, mentorship, and commitment to academic excellence have been invaluable. I also thank Associate Professor Petra Urbanová for her valuable advice and seamless collaboration, which connects our research with anthropometry.

For this work, I also utilized facial data from my colleagues and friends, specifically Sanggeta Biswas, Josef Hájek, Ivo Juříček, Gabriela Nečasová, Lukáš Novák, and Petra Snášelová, for which I am thankful.

I would additionally like to thank my colleagues who contributed ideas and suggestions that led to the improvement of this work. Special thanks to Štěpán Rydlo for providing technical support.

I would also like to express my gratitude to my friends and family for their unwavering support, encouragement, and understanding during the ups and downs of this challenging endeavor. Special thanks to Martina Grzybowská, who provided a different perspective on the thesis and helped improve it.

During my PhD studies, I supervised approximately 40 bachelor's and master's theses, with many of my students achieving interesting results that expanded my knowledge in the field of face recognition. Without these students, progressing in the research for this work would have been more challenging.

As I reflect on the 8 years of my PhD studies, I must acknowledge that it was an interesting experience that brought me a wealth of knowledge and introduced me to many fascinating individuals. In conclusion, I would like to express my gratitude to everyone not mentioned above who had a positive impact on the completion of this work.

Contents

List of Figures

Chapter 1

Introduction

Recognizing objects, or, more specifically people, through visual perception stands as one of the fundamental abilities of the human brain. Although we perform this task effortlessly on a daily basis without conscious effort, replicating this capability through computational methods has proven to be a formidable challenge. For a long time, the presence of a human was essential in situations where person identification was required. However, research has made significant progress since those early days, evolving the algorithms from rudimentary techniques focused solely on face recognition, to the development of fully autonomous recognition systems.

Compared to other biometrics used for identifying a person, face biometrics is one of the easiest to obtain - one just needs to capture an image of the person's face, whereas other biometrics usually require special sensors and, most importantly, the cooperation of the person. This proves to both a great advantage and a major liability - government agencies can perform recognition of people using today's extensive network of public security cameras, which can contribute to public safety, but the same approach can be misused to track people for malicious purposes.

In the first part of this thesis, state-of-the-art methods for biometric face detection and recognition are investigated. The primary emphasis of the research is on methods for generating a dataset from scanned 3D head models of real people. Subsequently, this work shifts its focus to the use of face detection and recognition algorithms on embedded devices. This includes both detecting the presence of people in images and evaluating the recognition of their identities. The use of such devices can offer improved privacy protection without sacrificing personal identification capabilities.

1.1 Research questions

In this thesis, we explore the available methods for face detection and recognition, which also bring observations useful for forensic face examination. With the gained knowledge, we perform experiments using state-of-the-art algorithms on both publicly available and our semi-synthetic datasets. Using the results of our experiments, we propose modifications of the selected algorithms with the aim to use these algorithms on embedded devices. The diversity within the domain of face recognition and analysis research topics has led us to outline the specific objectives presented below.

We have defined two research goals for this thesis. In general, for the training and evaluation of a neural network, it is important to secure an appropriate dataset. As a part of the first goal, we explore ways to generate a semi-synthetic dataset with face images.

The second goal is to develop an embedded device to execute the face recognition process using accelerators suitable for embedded devices.

The following two goals of this dissertation thesis have been defined:

- **Generated dataset for evaluation of face detection and face recognition** - due to the need to obtain a dataset to evaluate the influence of head (face) rotation on face recognition performance, we propose and implement a pipeline to generate a dataset with the ability to preserve the facial characteristics of real people. Furthermore, we compare the generated dataset with a dataset containing real images.
- **Propose and implement a device for face recognition using neural network accelerators** - nowadays, the available technology makes it possible to create smart cameras to perform face recognition. However, in general, face recognition intrudes upon a person's privacy. By utilizing an *embedded device* also called *edge device*, we can eliminate privacy concerns. For the proposed device, we modify the available state-of-the-art algorithms in order to adapt them for use on embedded devices.

Chapter 2

Introduction to face recognition and face characteristics

People recognize each other by specific criteria that belong to face recognition. In addition, some features of the face can help to determine the emotional or health state of a person. The use of facial appearance finds application through fields that commonly affect our lives. The contribution of research focused on face recognition and face analysis using machine learning methods is significant for security purposes. Unfortunately, except for positive contributions, the face recognition system can also have a negative impact on life due to privacy violations, identity theft, and other aspects raise privacy concerns [1]. On the other hand, with the face recognition systems that can quickly identify a person, we can save people's lives by the possibility of finding a concrete person quickly or suppressing a crime quickly.

In general, face recognition falls under the umbrella of biometric systems and computer vision, which are designed to process biometric characteristics to determine identity. On the other hand, the methods assigned to face analysis can be utilized in various fields such as criminology, healthcare, or consumer sector. In the following section, we try to familiarize the reader with information to better understand the issues related to face analysis and face recognition. Let us begin with an introduction to the field of biometrics.

2.1 Biometric systems and face recognition

Biometrics originated from the combination of two Greek words, *bio* (life) and *metrics* (measure) [2]. *Biometric characteristics* [3] have been used for several centuries. Interestingly, before 500 BC, Babylonian merchants used fingerprints as part of their records of commercial transactions. After the industrial revolution, there was a development of systems that work with other biometric features. Today, we encounter biometric systems on an almost daily basis.

We have mentioned the term *fingerprint* [4][TG.1], which represents one of the most widely used biometrics. However, it is not the only one. Other anatomical characteristics used in biometrics include the iris, retina, finger and hand veins, DNA, behavioral characteristics, and, of course, the face. Each of these biometrics has its own advantages and disadvantages, which arise from their underlying biological nature and depend on the specific use case. For each biometric, we assume that it possesses a certain amount of entropy information, which has been determined and quantified through biometric research. Fur-

thermore, when selecting an appropriate biometric system, we need to consider several aspects such as the level of unambiguity, resistance to imitation, inter-class and intra-class variability, and stability over time [4][5]. *Intra-class variability* [6] refers to the degree of variation between samples within the same class, while *inter-class variability* [6] determines the degree of variation among samples from different classes. Based on these considerations, we should select biometric properties with minimal intra-class and maximal inter-class variability. This selection ensures the ability to effectively discriminate among different samples.

In practice, the most commonly used biometric systems are based on fingerprints, iris patterns, hand shape, and face images (2D and 3D) [7]. Fingerprint biometric is the most widely used biometric that has penetrated various areas of public life. At the goverment level, fingerprints are used in police criminal investigations, in connection with biometric passports, in asylum procedures, and in many other areas. In terms of the consumer sector, they are most commonly used to secure laptops and mobile devices. A disadvantage is the moderate universality of fingerprints and the possibility of theft of this biometric data (obtainability from various surfaces) [4].

Biometric features are used in biometric systems to verify identify and finally authenticate a person. The first type of identity recognition is *identification* [6], which expresses a process to determine the person's identity. The results can be used to determine a person's identity from a reference set of identities for N subjects. From this perspective, identification is also known as 1:N matching. While the second type of identity recognition is *verification* [6], unlike identification, verification involves comparing one identity with another to determine whether or not the identities are the same.

The biometric system [8] consists of several interconnected modules (Figure 2.1). The first module is responsible for capturing a user's biometric characteristic (1), whereas one or more technologies can be employed. This is especially true for fingerprint sensors. To capture a face, color cameras are usually used. The two subsequent modules of a biometric system carry out preprocessing (2) and feature extraction (3). The objective of preprocessing is to clarify the biometric characteristics by suppressing all unrelated elements. Consequently, the biometric features are mapped to a specific format using a feature extractor, which is the holy grail of the biometric system. Although the variability of the individuals is primarily derived from biological factors, the extractor should be designed with respect to maximizing inter-class variability and minimizing intra-class variability. Furthermore, the algorithm should be designed with an emphasis on extracting features that have discriminative ability. Such features are processed by a template generator to map the features into a suitable format that allows them to be stored (5) and compared (4). The module responsible for template comparison takes an input feature template and, then, depending on whether it works on the identification or verification, it selects one or more templates from the data storage, and performs a comparison (6). However, it is impossible to declare two different biometric templates as equal with absolute certainty. Therefore, a score is assigned to each comparison to express the degree of similarity. The outputs of matcher are then provided to an application (7).

In terms of security, biometric systems are burdened with many vulnerabilities, therefore, their security is a crucial part of their development. The next factor that should be considered is the processing time. When a biometric system is used as an access system, fast processing of the biometric characteristic is required. However, there might be a trade-off in the preprocessing or feature extraction stage, which may negatively affect the performance of the whole system. In general, the developer of the biometric system must balance these

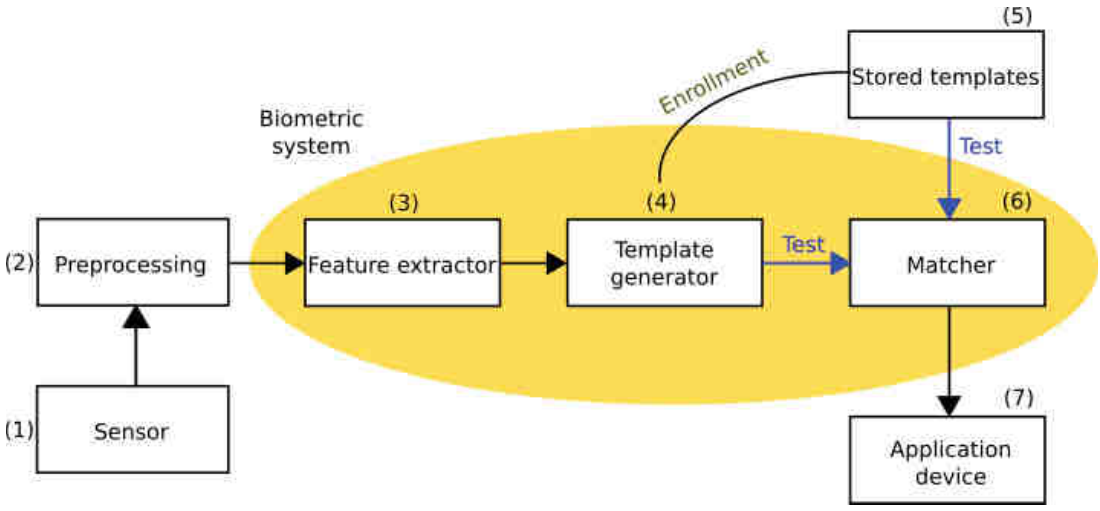


Figure 2.1: Simplified diagram of a biometric system. Adapted from [9].

factors. These shortcomings can be overcome by using domain-based hardware to accelerate the algorithms associated with the biometric pipeline, such as a neural network accelerator.

The biometric system that allows the processing of more than one biometric characteristic is called *multimodal system* [10][6]. An example of such a device is a sensor that works with fingerprint and face images to construct a robust biometric template that can be created by fusing two sets of the features. Such a system has better ability to deal with presentation attacks or a biometric characteristic damaged by a disease, compared to a biometric system utilizing only one characteristic. Overall, the advantage is the greater degree of reliability of such systems.

In addition, user privacy is the next aspect that is closely related to biometrics. Due to the nature of biometric data, there are legal and privacy issues that need to be considered when deploying or developing a biometric system.

2.1.1 Face recognition

The goal of the algorithms used in biometric systems that work with face images is to extract significant information from the face and create features from them. The input data for these systems are primarily image data composed of the color components visible to the human eye.

The first documented use of face recognition in legal proceedings was reported in the British court in 1871, where a face image was used to identify a subject [11]. In terms of the relationship between face recognition and computational means, the first semi-automated process was presented in 1964 [12]. From a technical perspective, the system was based on measurements between selected facial points. The set of measurements was then uploaded to and processed by a computer. This means that the computer only worked with a set of values that had to be measured by a person. In the following years, this method was expanded to include new features such as lip width and hair color.

With the advent of statistical methods, the next era in face recognition history began. In 1991, the method based on *Principle Component Analysis* (PCA) [13] was introduced by Alex Pentland and Matthew Turk. In contrast to the previous approaches, an input face image is directly mapped to a specific numeric representation. In 1998 [14], the Defense

Advanced Research Projects Agency (DARPA) introduced a program to support the development of face recognition algorithms. Within the program, a challenge dataset consisting of 2,400 images from 850 subjects was encompassed.

The expansion of the *Closed-Circuit Television* (CCTV) system has presented opportunities to deploy systems to automatically solve certain tasks. In the 1990s, Great Britain, especially London, was the leader in using the CCTV systems for surveillance. For example, an automatic plate recognition system was deployed in London in 1997. Among other approaches, a pedestrian tracking system can be included; historically, this solution was based on the method of background subtraction, which allows to extract a mask of all objects in the image. Subsequently, by storing individual masks over time, we can track the movement of individual object silhouettes. This principle is one of the possible approaches to object tracking. Another algorithm for tracking multiple individuals in an image using Gabor wavelets has been published in [15]. In general, the first system capable of recognizing faces was put into practical use in London in 1998 [16].

The "golden age of face recognition" came after 2001 when the World Trade Center in New York was attacked. This attack changed the world in many ways, and new requirements for international security emerged. As a result, face recognition systems were rapidly deployed at international airports. Of course, the use of these systems raises privacy issues.

The latest breakthrough in face recognition occurred in 2014, when Facebook unveiled FaceNet [17]. From that moment on, face recognition has used neural networks.

Available approaches

In the previous section, we defined face recognition as a process that uses a face image as input, which can be obtained from surveillance cameras, cameras in access sensors, or commonly used cameras, i.e. webcams, mobile phone cameras etc. Over time, methods have arisen from various principles, which has led to the categorization of these methods. While studying the literature, we were surprised that the categories of the approaches were different through the literature and papers. Based on [18][19], we divided the approaches into the following categories according to the milestones which are marked in Figure 2.2.

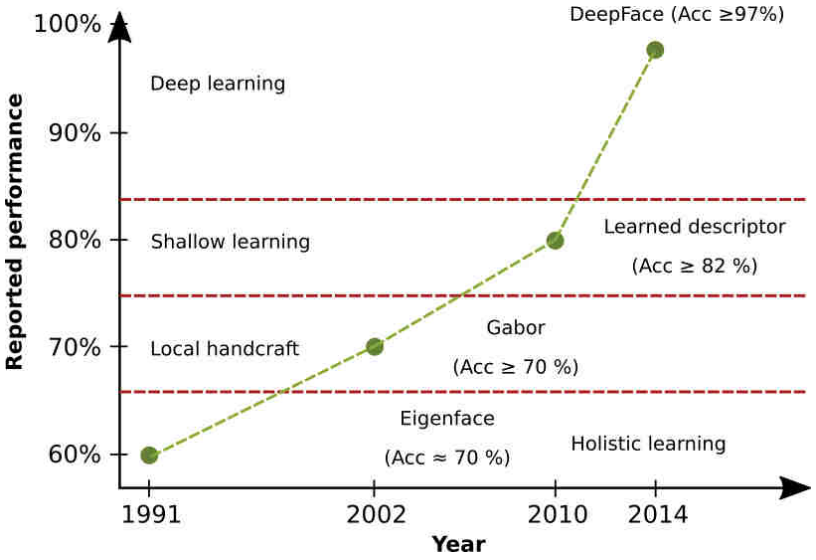


Figure 2.2: Timeline of the major milestones in face recognition. Adopted from [19].

Holistic methods - In the 1990s, the field of face recognition primarily relied on holistic methods. These methods were characterized by their ability to extract information from the entire image. This was achieved by mapping the input image to a subspace. Given the knowledge of linear algebra, we can reduce the space defined by M images with N×N to a subspace that is represented by a non-singular basis. Holistic methods can be further categorized into linear and non-linear approaches. In the case of linear methods, we assume a direct mapping to the subspace by a linear transformation. Notable representatives of this category are methods based on PCA, as discussed in Section 2.2.2, as well as methods derived from *Linear Discriminant Analysis* (LDA)[20], as discussed in 2.2.2. The non-linear subcategory encompasses methods that process an input image map using a kernel. For classifying face images, the transformation matrix is employed to map the input image to the subspace, followed by computing the distances to all classes associated with that subspace.

Local-feature based methods - in the early 2000s, the next generation of algorithms began. Unlike holistic approaches, these methods deal with the local appearance of a face image. For their detection, filters and detectors with the ability to capture a required part of an image were used. The methods are called *handcraft methods*, because the "by hand" preparation of essential filters is their crucial part. Then, the local features can be described by histograms, geometric properties, pixel orientation, or other forms to describe the data for machine learning processing. Similar to the previous category, these methods are categorized into the local-appearance based methods and keypoints based techniques. This task can be solved using pattern recognition algorithms, while their advantages include such as higher invariance to low light, variance in brightness, and resistance to monotonic gray scale changes [21]. For example, to create local histograms, the face image is transformed into a suitable form using the *Local Binary Pattern* [22] algorithm. In addition, we can describe the face using the features obtained by using *Gabor filter* [23].

The second subgroup utilizes the facial keypoints to create a descriptive template to represent the face. The term facial keypoints refers to well-defined points in the surface of the face that may be related to face anhtropometry, see Section 2.1.2. In addition to facial keypoints, it is possible to describe faces using Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) points [24].

Method based of shallow learning - in the 2010s, there was another change in the history of face recognition. The advent of learning-based local descriptors opened up a new way to describe faces [25]. However, these features do not consider all essential characteristics of a face. Moreover, the published approaches were unsuitable to be used in the real conditions. In our view, this milestone was the first step in the major expansion of deep learning.

Deep learning method - although the research community had explored various methods to improve individual steps of face recognition, these approaches have not been able to cope with the variability in the conditions under which the face can be captured. The breakthrough came in 2014, when Facebook introduced a novel solution called DeepFace. This was the first time a computer algorithm achieved human-like recognition performance. Since then, the vast majority of researchers have adopted deep neural networks as the basis for their solutions.

Although many algorithms for face recognition have been introduced in the last decades, we decided to describe only Eigenface and Fisherface from the older algorithms, see Sections 2.2.2 and 2.2.2, respectively. The best way to perform face recognition is to use a deep neural network. The theory of neural networks is described in Section 3.5.

Commercial software

Just as there are open source solutions available, there are many commercial applications designed for household to government purposes. For example, among the representatives of commercial software, can be included Clearview AI[1][2], Innovatrics[3], 3DiVi[4].

Challenges in face recognition

The design and implementation of robust face recognition algorithms is challenging because of the many factors that must be taken into account, such as rotation, lighting, image quality, and other factors. Most modern face recognition algorithms show high performance on a face image captured under controlled *"laboratory"* conditions. On the other hand, when the image is captured in a real world environment, commonly referred to as *"the wild"* [26] in the literature, recognition performance may be insufficient. In recent years, research has focused on improving the performance of such algorithms.

To date, researchers have found several ways to suppress the influence of these factors. Before a face image is processed by a face recognition algorithm, it is possible to improve its quality in the preprocessing stage. However, by adding another stage, the requirements to the computing systems may increase. Therefore, we must consider the longer processing time when designing biometric systems.

Another approach is to focus directly on the processing algorithm itself, which can be modified to improve tolerance for miscellaneous input variations. However, the approach must be developed with respect to a required use case. In the case of forensic applications, it is crucial to ensure that the properties in a face image remain unaltered.

Since modern approaches are based on neural networks that map a face image to a vector, it is beneficial to understand how this vector is affected by various conditions. With this knowledge, we can effectively address weaknesses in face recognition algorithms with the goal of suppressing or even eliminating them. Therefore, it is important to summarize information about known challenges in the field, such as variations in lighting conditions, pose, and occlusion.

For a better understanding, Figure 2.3 shows four faces and their corresponding embedding vectors (the process of generating embedding vectors is described and discussed in Section 3.5.1) visualized using heatmaps. The face images on the left side of the figure are taken in frontal position, while the upper right is a capture of the rotated face and the lower right face photo is related to another person. In general, a larger distance indicates a lower probability that the face belongs to the same identity.

Nowadays, the ability of the algorithm to deal with environmental influences is a major challenge. Therefore, we first focus on the environmental conditions relevant to the face image. According to [27], these factors include:

- *Light conditions* - the distribution of light in the whole image should be uniform, if not, it causes decrease of recognition performance.
- *Complex backgrounds* - due to the complexity of a scene, face detection can be affected.

[1] https://www.clearview.ai/

[2] Due to the violation of privacy law, the company is banned for selling its service and products to most US companies.

[3] https://www.innovatrics.com/

[4] https://3divi.ai/

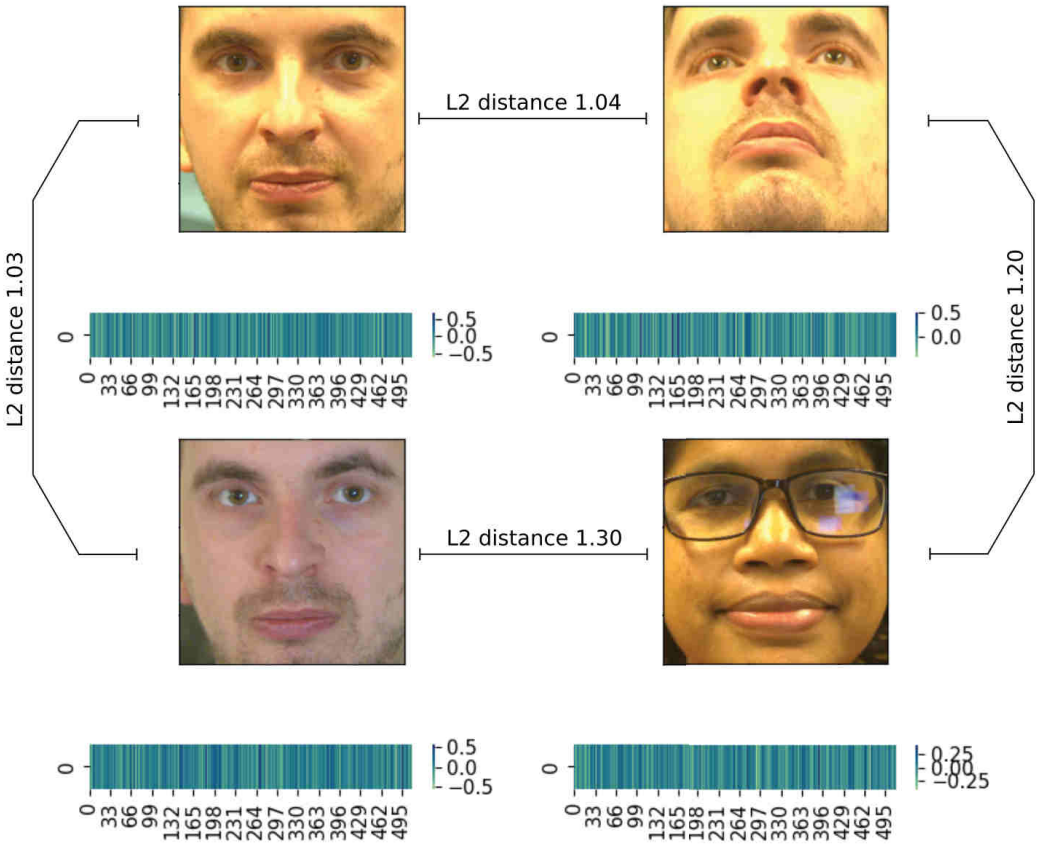


Figure 2.3: Face images and their visualized embedding vectors obtained by the ArcFace algorithm, for more information see Section 3.5.8.

- *Many faces in one image* - similar to the previous case, the presence of multiple faces in an image can also affect the recognition performance.

Another group of unfavorable conditions for face recognition arises from the face itself and these include [28]:

- *Pose variations* - rotation of the head can change the appearance of the face. To suppress this problem, a robust biometric template must be created.
- *Expression* - the variability in face expressions among the images of the subjects can aggravate face recognition.
- *Plastic surgery* - plastic surgery brings a greater ability to modify significant characteristics of a face.
- *Aging* - throughout life, the appearance of a face is affected by aging.
- *Occlusion* - similarly, facial accessories can contribute to the degradation of recognition capabilities.

The last group determines which properties of a camera have an influence on an image quality. These include the following properties:

- *Resolution* - for face recognition, it is essential to choose suitable resolution, for more information, see Section 2.3.2.
- *Image compression* - with the compression (i.e. JPEG), the small artifacts can appear on the face, which causes the worst performance of both face detection and recognition.
- *Distortion* - the face radial distortion can greatly degrade the performance of existing face recognition methods [29].

The next challenge lies in the biological variation of the anthropometric properties of the face involved and the variation in appearance, such as color and skin pigmentation. In 2020, there was a case in the USA where three dark skinned men were wrongly arrested due to a faulty evaluation by the recognition software [30]. In subsequent studies, the researchers focused on analyzing the effect of skin color on face recognition performance. The results confirmed that the performance of face recognition on people with dark skin is lower than that of people with Caucasian skin type. Buolamwini and Gebru [31] found that the dataset selected to train the face recognition algorithm was imbalanced, with a higher representation of Caucasian type identities.

2.1.2 Facial antrophometry

Anthropometry [32] is a scientific field that deals with the measurement of characteristics of the human body. Body characteristics are unique for each individual, and the same is true for the face. In the following section, we introduce the anthropometric subfield that describes the face using the knowledge gained from quantitative evaluation of human face morphology [33]. The importance of face anthropometry is given by the fields where it is used, which include forensic application, orthodontics, biometric system, and maxillofacial surgery in which it is an essential part.

In the past, facial features were only measured using specialized equipment, including anthropometers, personal scales, spread calipers, pelvimeters, sliding calipers, soft metric tape, and calipers [34]. Today, modern methods include computer vision algorithms with the ability to extract the facial landmarks from the face image or 3D face model, respectively. The essential part of the human face is defined by proportions that vary among individuals. The set of proportions is determined by researchers. For example, in the paper [35] 155 properties in a face were described, while in the paper [36] only 30 significant properties were selected. The evolution of humans has been diversified according to a branch of evolution that is specific to different places in the world. Therefore, there is no general set of face proportions that can be used to describe any individual face. Anthropometric researchers study the relationship between facial features, and this knowledge has a significant impact on face recognition. Therefore, face recognition algorithms must take this aspect into account.

Today, the work of anthropometric researchers can be facilitated by computer algorithms. These algorithms can automatically detect face regions and landmarks, which are then used to create an anthropometric description of the face. It is possible to capture more anthropometric points to create a more robust description when we have applications that can handle a 3D model.

Let us show some examples using face anthrophometry. In terms of forensic applications, most tasks are related to the identification of people. However, the identification of a dead person can be burdened by various forms of body decomposition. With applications, we can

perform body reconstruction based on its remains [37] and then provide information for crime investigation.

The second example comes from another field, specifically plastic surgery. Before the operation, it is necessary to determine which methods will be used for the operation, and it is necessary to specify when the operated face parts will move. In addition, with the pre-surgery position of face features, we should modify the statistics of these features. It means that parameters such as standard deviation or variance are changed to the given distribution of characteristics. It leads to refinement of statistics.

Finally, we came to an anthropometric perspective on face recognition and biometrics. Biometric recognition capabilities depend on the appearance of the face and its representation. Using a 3D face model and its texture, we should be able to extract more information than is included in a 2D face image. For face recognition, the characteristic part of the face is used in algorithms for face alignment, and in earlier approaches, called the feature base, was also used for face recognition. Similarly, the selected face characteristic should be used to determine the face rotation and so on for face classification. In addition, face anthropometric properties can be used to evaluate the quality of a portrait photo when capturing a face.

In the following section, we describe the essential facial characteristics and evaluate their possibilities of use in the face recognition process. According to [35], we know that the face can be described by up to 155 cranio-facial anthropometric proportions. However, the number of usable features depends on the use case. In face recognition, two key points are sufficient for the basic alignment that is needed for face recognition. On the other hand, face reconstructive and cosmetic procedures can be planned using all the craniofacial anthropometric proportions.

Face anthropometry landmarks and measurements

Next, we become familiar with the location of common facial landmarks used in anthropometry. Due to the negative influence of the landmark location on the accuracy of facial measurements, it is crucial to ensure their localization.

Besides the manual selection of landmarks, they can be localized by computer vision algorithms [38]. One of the possible ways to ensure accurate localization is to utilize the 3D model of the head [39]. However, in many cases such a model is not available. The goal of a face landmark detector is to detect the landmarks in a face image. Often, a face image suffers from lack of data to determine some landmarks. For better understanding, the following illustration (Figure 2.4) shows individual landmarks in the face.

As you can see, the number of landmarks is relatively large. Thus, we have focused on only a subset of them that have been identified through analysis of available studies. The following list is composed of selected landmarks and their description is according to [41].

- *Endocanthion* (en) - the point located at inner commisure of the eye fissure.
- *Exocanthion* (ex) - the point located at outer commisure of the eye fissure.
- *Subnasale* (sn) - the middle point of the angle at columella base where the lower border of the nasal septum and the upper lip meet.
- *Naison* (n) - the middle point of the nasal root and the nasofrontal suture.

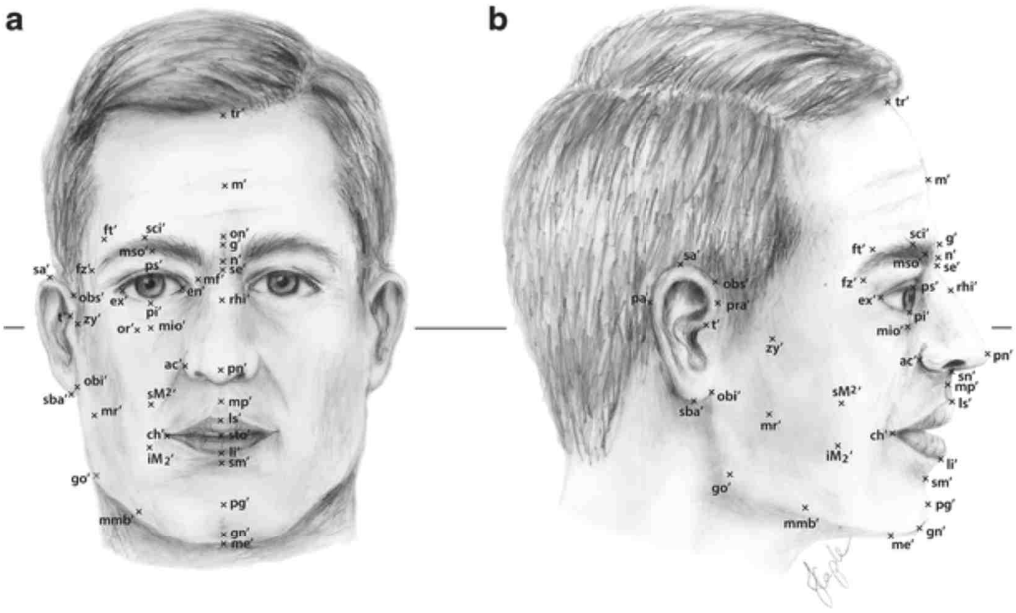


Figure 2.4: Facial landmarks in (a) anterior view and (b) right lateral view [40].

- *Gnathion* (gn) - the lowest midpoint landmark on the lower border of the mandible.
- *Cheilion* (ch) - the point at each labial commisure.
- *Tragion* (t) - the tragion is localized on the ear and their exact location is in the notch above the tragus.
- *Alare* (a) - the most lateral point on each alar contour.
- *Gonion* (g) - the most lateral point on the manidublar angel.

Consequently, the points are utilized to define the distances to describe the characteristics of a face. However, only a subset of these points is used in face anthropometry. As mentioned above, the location of each anthrophometric point is very well defined. Then, the measurement among them can be performed to determine a set of measurements with different degrees of variation within the human face per individual. Moreover, the statistical parameters of the measurements are affected by many factors, such as sexual dimorphism, ethnic variations, deme group variations, and others. Apart from the above mentioned possibilities of using facial measurements, we can estimate human age by combining them with the position of landmarks [42].

We have selected essential measurements [43] that can be used as a baseline for face analysis, face alignment, or face recognition:

- Forehead height (tr-n)
- Physiognomical face height (tr-gn)
- Morphological face height (n-gn)
- Face width (zy-zy)

- Mandible width (go-go)
- Mouth width (ch-ch)
- Nose height (n-sn)
- Forehead height (tr - n)
- Lower face height (sn-gn)
- Intercanthal distance (en-en)
- Biocular width (ex-ex)
- Eye fissure length (en-ex)
- Mandible width (go-go)
- Morphological nose width (al-al)

We intentionally did not include the pupillary distance. This measure is described in more detail below. The measurements allow us to classify the face according to the indices. These indices are defined by the ratios between the selected measurements. Among the indices is the *Facial Index* [44], which is used to assign a face to one of the five categories defined in Table 2.1.

Table 2.1: Categories of Facial Index (FI) [44].

Facial Index (FI)	
FI range	Scientific term
< 79.9	Hyper euriproscopic (very broad face)
$80 - 84.9$	Euriproscopic (broad face)
$85 - 89.9$	Mesoproscopic (round face)
$90 - 94.9$	Leptoproscopic (long face)
> 95	Hyperleptoproscopic (very long face)

Then, the part of identity represented by the face is derived from the measurements with large variability. This is best described in the following illustration (Figure 2.5), which shows which parts of the face have a strong influence on identity.

Overall, pupillary distance is the most widely used metric for 2D face recognition. In contrast to early feature-based approaches, modern approaches to face recognition primarily use the holistic view of a face image without focusing on individual facial features. For 3D face recognition, however, the combination of anthropometric features can be used [45].

Pupillary distance

With the exception of 3D face recognition, face recognition used by police, surveillance, and citizen identification systems uses a face image as input. In order to use a face image, it is necessary to ensure the required size of the image and also the orientation of the face.

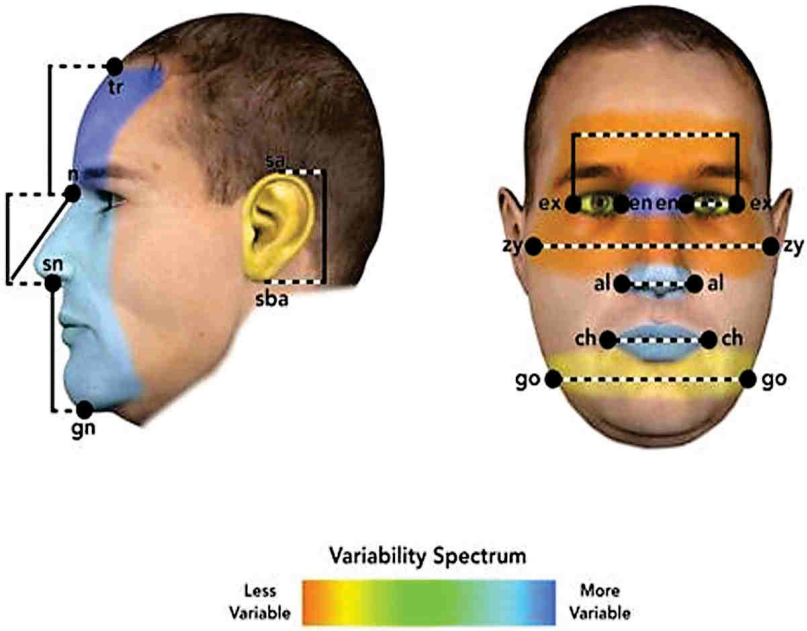


Figure 2.5: The extent of variability in face regions presented in a color spectrum [43].

For both, the position of the eyes can be used. The distance defined between them is called *Pupillary Distance* (PD) [46].

The pupillary distance is defined as the distance between the pupils (centers) of the eyes. According to the Anthropometric Survey of US Army Personnel [47], the mean PD for males is 64.0 mm with a standard deviation of 3.4 mm and for females the mean of the distance is 61.7 mm with a standard deviation of 3.6 mm. The study is unique in its inclusion of various ethnic groups of subjects. Although the study is primarily related to the Caucasian ethnicity (64.85 % of the evaluated samples), it also includes other ethnic groups.

Today, from the perspective of face recognition and analysis algorithms, PD can be used anywhere in the face recognition pipeline, which includes training, algorithm evaluation, identity inference, and even face classification using this metric. In the case of training and inference, the metric is used to align a face image in the roll rotation. Practically in all areas, this metric can be used for the normalization of faces on the basis of their size.

However, the alignment of the face image in the roll based on the eyes is limited. In addition, it is necessary to use another landmark for complete centering. This alignment method is also suitable for filtering the face image by resolution when it is obtained during the recognition process. For example, if its resolution is less than 16×16 pixels [48], face recognition should not be performed due to insufficient resolution. To suppress the deficiency, the image resolution can be increased using methods that employ interpolation or methods based on neural networks.

In another perspective, the PD metric can be used to identify the health disorder. For example, it includes near point convergence and stereo acuity [49].

Frankfort horizontal

The above sections describe the possible 2D alignment of the face. The most common way to perform anatomical alignment of a 3D head model is to use the *Frankfort plane* or *Frankfort horizontal* [50]. Mathematically, three points are needed to define a plane. In terms of the Frankfort horizontal alignment, the plane passes through the two anthropometric points located at the lowest point of the eye orbits and the third point outside the face, which is defined as the superior point of the external auditory meatus.

In a face image, we can determine the Frankfort plane by locating the anthropometric points that include the lowest points of the orbital ridge and the tragion point.

2.1.3 Security and privacy questions

In general, biometric systems are associated with privacy and security because of using and collecting personal information. Likewise, face recognition must address security and privacy. Unlike a fingerprint recognition system, a face recognition system processes a face image, which is an easily obtainable biometric feature.

In addition, the use of face recognition systems in government organizations and the consumer sector has expanded widely. In order to cover crucial aspects related to security and privacy, the conditions and laws are divided into levels, from general regulation to specific regulation. By the term general regulation, we mean the Law on Protection of Personal Data with laws related to biometric systems, while the specific regulation includes regulations for the specific case of using face recognition, e.g. surveillance face recognition systems. The issue of privacy protection in surveillance systems should be carefully considered, as the privacy of people in public can be seriously violated. In general, face recognition is considered to be the most "privacy-invasive" biometric technology than any other [51].

The first international convention aimed at ensuring data protection was presented in 1981, such as the Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data (ETS No. 108) [52]. The main focus of the Convention is to ensure the rights of individuals, in particular their right to privacy, regardless of their nationality or where they live. Simply put, the Convention defines laws for the collection, processing, and storage of personal data for participating parties. According to the Convention, digital biometric data is classified as data that uniquely identifies an individual. By processing a face image with technical means, we can uniquely identify or authenticate a person; therefore, the nature of the image meets the definition of biometric data under the Convention.

The first version of the Convention was published before the emergence of new ways of processing and collecting biometric data. Therefore, the Convention was extended in 2018 and is referred to as 108+ [53]. Today, the Convention has 55 signatories, 8 of which are not members of the Council of Europe (CoE). In the field of face recognition, the guideline on face recognition adopted by the Convention was introduced in 2021 [54]. The document consists of three parts covering the technology of face recognition. The first part is intended for legislators and decision makers, which defines the lawfulness and also determines the restrictions for state authorities.

Whereas the second part is related to providers and developers of face recognition system, and last, the third part is intended for entities using face recognition technologies. In general, the last part summarizes the rights of subjects that are provided in Article 11 of Convention 108+. In addition, it is mentioned that under what conditions the rights of subjects can be restricted by authorities.

In 2018, the next set of data protection rules, known as the *General Data Protection Regulation* (GDPR) [55], was introduced. This regulation is based on the standards and legal recommendations specified in extension of convection 108. In addition, the GDPR brings new requirements to the subject privacy file. Convention 108 is sometimes referred to as the mother of the GPDR.

In the USA, for face recognition there is established law called Biometric Information Privacy Act introduced in 2008 [56].

2.2 Historical methods for face detection and recognition

In the following section, we present methods for face detection and recognition that were commonly used around 20 years ago. Due to their earlier significance, we decide to discuss them in more detail.

2.2.1 Face detection

In the previous section, we emphasized the importance of face detection as a crucial step in the face recognition process. Assuming that the input image contains only one face, this step should not be performed and should be skipped. Historical face recognition algorithms operated on manually cropped and processed face images. Previously published face detectors solved face recognition as a binary classification, determining the presence or absence of a face. As a result, these detectors relied on machine learning classifier algorithms. However, the advent of neural network-based detectors has revolutionized the field, rendering previous approaches obsolete. In the next section, the principles behind these earlier detectors are discussed.

Viola-Jones detector

The Viola-Jones detector [57] marked a breakthrough in the field of face detection. It was published in 2001 and named after its authors. This detector offered indisputable advantages, including an excellent balance between computation speed and computing power requirements, as well as high detection accuracy. To detect faces, the detector utilizes a carefully selected set of weak classifiers with the ability to identify facial features.

In the case of binary classification, a weak classifier determines its outputs in a range close to the threshold. For example, if we consider a positive classification represented by one and a negative classification represented by zero, the weak classifier will provide a value close to 0.5 [58]. The weak classifiers used in the Viola-Jones approach are represented by Haar-like features derived from Haar wavelets [57].

In [57], the author's research identifies three types of features: edge features, line features, and four-sided features. Edge and line features are used for extracting edges and lines in an image, respectively. The last one is specifically designed for the extraction of the diagonal features.

During face detection, it is necessary to perform calculations for features responses. The Viola-Jones algorithm employs a different representation of a face image, called integral image, which enables a fast computation of the Haar features. The integral image is calculated only once, and the values of the features are computed by referencing the corners of specific areas. This allows us to efficiently calculate the sum of these areas, pro-

viding the exact information needed for further processing. Together, the integral image is calculated by

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i\left(x', y'\right) \tag{2.1}$$

where ii represents the integral image and i represents the input image.

The algorithms generate a large number of candidate features that can be counted in the thousands. However, only a small subset of these features is crucial for accurately identifying a face. Therefore, it is necessary to reduce the number of parameters and identify the subset of features that can detect faces in an image. The selection of appropriate features is done using the AdaBoost algorithm [57]. This algorithm aims to find a set of features that produce strong responses when a face is present in the image.

Furthermore, the algorithm selects approximately 2,500 features that need to be computed for each region in the image, which can be a time-consuming process. To address this, the authors employ cascading features systems, where the features are ordered from the most influential to the least significant. The best features are those that have a significant impact on detection. For example, this could include features that detect essential parts of the face, such as the nose, mouth, or eyes. By implementing cascade systems, the face detector can achieve real-time performance for face detection. Using a *sliding window* [59] approach, we traverse the image and perform the cascade classifier at each localization. At each level of the cascade, the output determines whether a face could potentially be present in that region. If a weak classifier provides a negative answer, the evaluation is terminated and the window moves to the next location.

On the other hand, if the response is positive, indicating a potential face *"maybe it is a face"*, the classifier proceeds to the next level of the cascade with the same behavior. Only when the last level of the cascade is reached, and it gives a positive response, it is determined that a face exists at the given location.

Detector based on Histogram of Oriented Gradients

Previously, one of the most famous detectors was a detector based on Histogram of Oriented Gradients (HOG) [60]. While Viola-Jones performs recognition based on the responses of Haar-like features, the HoG utilizes edges contained in the face area which are expressed by the histogram of oriented gradients. Detection in the image is performed by moving a sliding window over the image with a well-defined step and overlap.

The feature descriptor is calculated in several steps. According to the name of the algorithm, the essential step is the calculation of oriented gradients represented by angles and magnitudes. This is performed on each channel of the RGB or grayscale input image by convolution operation with Sobel filters for the x and y direction separately. The result is then represented by two resulting matrices with derivatives per pixel. Given by Eq. 2.2, the magnitude matric is calculated for each location defined by the coordinates x and y [61].

$$m(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} \tag{2.2}$$

where $f_x(x,y)$ is the component for the x direction and similarly the $f_y(x,y)$ is the component for the y direction. Furthermore, the angle is calculated using the following equation (Eq. 2.3):

$$\Theta(x, y) = \tan^{-1} \frac{f_y(x, y)}{f_x(x, y)} \tag{2.3}$$

where $\Theta(x, y)$ is the angle.

Furthermore, the matrix is organized into units called cells. For each cell, the histogram is computed by assigning magnitude values to bins according to their directions. However, the contribution of the magnitude depends on the directions, proportionally, the magnitude is divided between two closest bins depending on the direction of the derivative. In the published paper, the angular range 0-180° is divided into nine bins.

Optionally, we can enhance the histograms in cells by block normalization with the aim of ensuring the brightness invariation. Normalization is performed by concatenating the histograms in a given block and calculating the normalization factor, which is then divided by the values of the individual histograms. This is performed on blocks over cells. This gives the resulting descriptor higher resistance to changes in illumination and shadows.

In the last step, individual histograms are concatenated to one vector, which is used as a descriptor.

Consequently, the descriptors are used as inputs to the classifier to determine the class probability. For this purpose, we can use *Support Vector Machine* (SVM) [60]. Note that the SVM is primarily designed for binary classification. However, by using the kernel to map the input to the subspace, it is given the ability to solve a multiclass classification.

2.2.2 Face recognition algorithms

The face recognition algorithms published in the 1990s were based on linear algebra methods. With the expansion of neural networks, these algorithms are already being outperformed. However, the algorithms essentially use techniques from the field of machine learning. Familiarity with these techniques is essential for face recognition. Anyone seeking a comprehensive understanding of face recognition should be familiar with at least two important algorithms, such as *Eigenface* and *Fisherface*. In the following text, we explain the principle of these algorithms.

Eigenface

The Eigenface was introduced in 1991 by Mathhew A. Turk and Alex P. Pentland [13]. In general, the faces represented by images share a set of features that are common to all faces. For example, the face of a healthy person includes mouth, eyes, and nose. Face images may also contain other less obvious features and noise. At the training time, the Eigenface algorithm views the images as a set of vectors. This set of vectors is represented in huge multidimensional space, making it difficult to extract useful information directly. Therefore, it is necessary to reduce the dimensionality of the space. To do this, the Eigenface uses the *Principal Component Analysis* (PCA) [62] algorithm, which is commonly used in machine learning.

From an abstract point of view, the Eigenface maps the feature space into the principal components, called the *eigenspace*. PCA then determines a correlation matrix, which is used to calculate the eigenvectors and their values. In linear algebra, the eigenvector is a nonzero vector such that it does not change direction when the linear transformation is applied to it. When the eigenvector is multiplied by a scalar, the magnitude of the vector can be changed while the direction remains the same. The scalar is then called an eigenvalue.

Let us see how PCA reduces an input space to a subspace and how the eigenvectors are determined. Based on [62], the training is divided into the following steps.

1. *Dataset preprocesing and calculating covariance matrix* - the first step is to unfold each image from the two-dimensional matrix ($N \times N$) to the vectors Γ with length $n = N \cdot N$ and create the set S of image vectors $\Gamma_1, \Gamma_2, ..., \Gamma_M$. Subsequently, the images are centered by subtracting their mean, computed as follows (Eq. 2.4)

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i. \tag{2.4}$$

Then, the centering is performed by subtraction Ψ from each image represented as the vector Γ. We obtaine a new set of vectors represented by the matrix A (Eq. 2.5) [62].

$$A = [\Phi_1, \Phi_2, \ldots, \Phi_M]_{n \times M}. \tag{2.5}$$

where $\Phi_i = \Gamma_i - \Psi, i \in \mathbb{N}, 1 \leq i \leq M$. Now we determine the covariance matrix, intended for dimensionality reduction, defined by

$$C = \frac{1}{M} \sum_{i=1}^{M} AA^T \tag{2.6}$$

where A is the matrix composed of the image vectors. Assume that the size of C is $M \times N$, where M is the length of the vector and N is the size of the vector. This may cause the computational effort to be high and thus the covariance matrix to be difficult to compute. If $N << M$, then it is advantageous to compute the covariance matrix by $C_r = A^T A$. In [62], there is proven that the eigenvectors of the covariance matrix C are the same as the eigenvectors of the second covariance matrix C_r.

2. *Find eigenvectors and eigenvalues* – the next step is to find the corresponding eigenvectors and eigenvalues in the covariance matrix. The orthonormal basis of the subspace is defined by eigenvectors with non-zero eigenvalues. According to the eigenvalues, the eigenvectors are sorted in descending order. The eigenvector with highest eigenvalue represents the great variance in the images. For each face, the weights corresponding to the set of the eigenvectors are defined. In this way, each face in the training dataset can be defined.

2. *Classification* - during the recognition process, the input image is converted into a vector and the mean is subtracted from it. The vector is then projected into eigenspace and assigned to a face class based on the projection weights.

Fisherface

Currently, we know that the Eigenface projects data by eigenvector in the direction that has the most variation. However, the labels of the data samples are not taken into account when calculating the eigenvectors. Given this fact, the weakness of Eigenface lies in its ability to effectively separate different classes. Under ideal conditions, including precise face alignment, homogeneous lighting conditions, and others, the classes are linearly separable.

Nevertheless, most face images are captured in the wild under conditions that do not meet the requirements, see Section 2.1.1 for more information. This leads to a large scatter of the vectors in space. As a result, the subset of samples becomes ineffective when PCA is applied.

Using the labels, we can design a more appropriate method to reduce the dimensionality of the space. The *Fisherface algorithm* [20] considers the class label to ensure the ability

to maximize the separability among classes. Mathematically, the essential part of the Fisherface arises from Linear Discriminant Analysis (LDA), and is known as Fisher Linear Discriminant (FLD). From a high-level perspective, the objective of FLD and PCA is the same, to obtain the projection matrix W to determine the eigenvectors and eigenvalues.

However, unlike PCA, the projection matrix is obtained from the class scatter matrix and the within-scater matrix consists of the sum of the scatter matrix for individual class. The main objective is to maximize the ratio between them.

The within-scater matrix is given by (Eq. 2.7) [20]

$$S_W = \sum_{j=1}^{C} \sum_{i=1}^{N_j} \left(x_i^j - \mu_j\right) \left(x_i^j - \mu_j\right)^T \tag{2.7}$$

where x_i is the i^{th} sample of class j, μ_j is the mean over all images in the corresponding class j, N_j expresses the number of samples in class j and C is the number of classes.

Whereas class scatter matrix S_B is calculated by the global mean μ and the individual class mean μ_j. In general, the equation is given by (Eq. 2.8) [20]

$$S_B = \sum_{j=1}^{C} N_i \left(\mu_j - \mu\right) \left(\mu_j - \mu\right)^T \tag{2.8}$$

.

where N_i expresses the number class of samples in class i. Consequently, the projection matrix W is computed to maximize the ratio between the determinant S_W and S_B given by the following formula, defined by [20]

$$W_{opt} = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} = [w_1 w_2 ... w_m] \tag{2.9}$$

where w_i is the i^{th} generalized eigenvectors of S_b and m expresses the number of these vectors.

By the prove [63], we assume that the ratio should be maximized when the projection matrix is non-singular. Such matrix consists of generalized eigenvectors i.e.

$$S_B \mathbf{w}_i = \lambda_i S_W \mathbf{w}_i, \quad i = 1, 2, \ldots, m \tag{2.10}$$

that is fulfilled provided that the maximum rank is $N - C$, where N denotes the number of samples, C denotes the number of classes and λ_i is corresponding eigenvalue to eigenvector w_i.

However, if the number of samples N is smaller than the dimension of the face image space (number of pixels per image), we have to assume that the matrix S_W may be singular. This makes the Fisherface calculation difficult, since the inverse of the matrix S_B is not possible. To deal with this problem, the paper [20] proposed a solution to overcome this problem by reducing the face image space to $N - C$ using the PCA algorithm. The modified project matrix, denoted as W_{opt}^T is defined as

$$W_{opt}^T = W_{fld}^T W_{pca}^T \tag{2.11}$$

where

$$W_{pca} = \arg\max_W \left|W^T S_T W\right|, S_T = \sum_{k=1}^{N} \left(x_k - \mu\right) \left(x_k - \mu\right)^T \tag{2.12}$$

$$W_{pca} = \arg\max_{W} \left| W^T S_T W \right| \tag{2.13}$$

$$W_{fld} = \arg\max_{W} \frac{\left| W^T W_{pca}^T S_B W_{pca} W \right|}{\left| W^T W_{pca}^T S_W W_{pca} W \right|}. \tag{2.14}$$

Overall, PCA is used to reduce the face space, while FDA is used to discriminate among classes. However, the Fisherface algorithm suffers from several problems. First, the algorithm is very complex and requires a lot of computing power. Second, similar to Eigenface, the accuracy of recognition is affected by the attributes of the face image.

2.3 Face recognition standards and performance metrics

The following section describes the standards related to face recognition. In general, standards play an important role in the development, deployment, and use of biometric systems. In addition, the performance metrics for evaluating the face detection and recognition system are also described.

2.3.1 Standards

Biometric standards have been published to ensure uniformity of biometric system evaluation metrics, methodologies, and terminology. As noted above, the use of biometric standards is critical to the evaluation and deployment of biometric systems from different vendors [64]. Likewise, rules and recommendations for collecting biometric data play an important role in biometrics.

In general, biometric standards are a collection of documents that aim to ensure interoperability and data exchange between biometric applications and systems, respectively. Such documents are maintained by international institutes with a large number of participants to maximize the likelihood of ensuring consensus.

The first attempts to define biometric standards were made around 2000 when the boom in biometric systems began. This led to the creation of the first international committee for biometric standards maintained by the International Organization for Standardization (ISO) established in 2002, ISO/IEC JTC[5] 1/SC[6] 37 [64]. The scope of the committee is focused on a general biometrics and its aim is to cover the following areas of standards that include (taken from [64]): common file frameworks; application of evaluation criteria to biometric technologies; biometric application programming interfaces; biometric data interchange formats; related biometric profiles; methodologies for performance testing and reporting and cross-jurisdictional and social aspects.

The development of the standards is being carried out by six working groups, including Harmonized Biometric Vocabulary (WG1), Biometric Technical Interfaces (WG2), Biometric Data Interchange Formats (WG3), Technical Implementation of Biometric Systems (WG4), Biometric Testing and Reporting (WG5) and Cross-Jurisdictional and Societal Aspects of Biometrics (WG6).

However, the committee has relations with other JTC 1 subcommittees. For example, the committee is involved in ISO/IEC JTC 1/SC 17 [65] and ISO/IEC JTC 1/SC 27 [66], the first covering cards and personal identification and the latter covering IT security

[5] Joint Technical Committee
[6] Subcommittee

techniques. During the development of a standard, the ISO committee collaborates with an external organization such as the International Labour Organization (ILO).

In total, the number of published standards covered by ISO/IEC JTC 1/SC 37 from its inception to 05/2023 is 136, with 20 standards under development.

Let us examine standards in greater detail. One of the most important sets of standards is the BioAPI intended for implementation of a uniform interface to ensure integration of different biometric modules using defined Biometric Interworking Protocol (BIP). The first version of BioAPI (BioAPI Specification Version 1.00) was introduced by the BioAPI Consortium in 2000. In addition, BioAPI v1.0 (BioAPI - ANSI INCITS 358-2002) has been revised and extended due to the creation of new standards registered under ISO/IEC JTC 1/SC 37. As a result, the standard is included in ISO/IEC 19784-1 [67].

From a technical perspective, the essential part of BioAPI is the BioAPI Framework, which is designed to communicate with Biometric Service Providers (BSPs). BSPs provide support for various biometric capture devices, preprocessing algorithms, matching algorithms, etc. In BioAPI v2, the foundation of the topology has been modified by adding the Biometric Function Provider (BFP). In the topology, the BFP is placed between the biometric application and the BSP. This change reduces the complexity of the implementation and leads to the implementation of the solution by the vendors. In addition, the BioAPI has defined the BioAPI Interworking Protocol presented in ISO/IEC 24708 [68] for communication among the different multisystems.

The next significant means to ensure interoperability among biometric systems is the use of a uniform data format designed to represent different data. This format is described in the ISO/IEC 19785 [69] standard and is provided by the Common Biometric Exchange Formats Framework (CBEFF), which defines approaches for sharing biometric data and effectively performing serialization. From a perspective of biometric systems, the standards introduced in 2006 offer new opportunities for exchanging data between systems from various vendors using standardized structures.

It would be a mistake to ignore the domain-specific standards associated with a biometric characteristic. This includes standards such as the Biometrics Data Interchange Formats [70] for face images, iris images, and fingerprint images.

Due to the topic of this thesis, we focused only on biometric standards related to face recognition. Capturing the face image can be affected by many factors. Given appropriate rules and recommendations, we should control some of them in order to achieve better quality of the face image.

The necessary step in the development of a biometric system is to test its performance or its ability to withstand attacks. This means creating well-defined scenarios and establishing a uniform procedure for testing the algorithms. This is the only way to compare them.

With the growing trend of using biometric technology, it was necessary to establish new standards and methodologies for testing biometric systems for mobile phones. From a user perspective, it was important to understand the reliability of the sensor in a mobile phone and the ability to secure the device. In response to this need, Google and the FIDO (Fast IDentity Online) Alliance [71] have introduced new standards related to biometrics in mobile phones. An important information from the Google standard [72] is that the biometric standards for Android (>11) consider face recognition as a weak biometric, compared to fingerprint recognition.

The requirements defined in the Biometric Data Interchange Formats for Face Image ISO/IEC 19794-5 [70] standard include specific scene, photographic, digitization, and format requirements for both human verification and face recognition by a computer program.

Even the process for capturing a portrait photo used by officers is defined by the standards. The appropriate quality of an ID portrait photo can be ensured by following the recommendations and requirements for capturing the portrait photo.

2.3.2 Portrait photo standard

Besides the view on face recognition as a machine learning task, we should not forget that a face image can be utilized to obtain identity by human evaluation. Because it is easy to compare a real face with a face image, the face image is often used on an ID card. Especially for official documents, it is necessary to ensure that a face image is captured according to well-defined requirements and recommendations. Therefore, one way how to achieve this goal is to use the rules of a standard.

The process of capturing the portrait photo follows a standard denoted ISO/IEC 19794-5 [70], the importance of which is emphasized by the adoption of the standard by the ICAO organization to establish rules for Machine Readable Travel Documents (MRTD) [73].

The standard has a direct impact on automated face recognition, although historically it has been used primarily for manual face verification. In Section 2.1.1, there is noted that the quality of a face image is affected by many different factors. Fortunately, in the case of laboratory conditions, we can suppress the influence of these factors more than in the "wild condition".

The following text summarizes important information that is essential to achieve the appropriate quality of a face image. Some parts are related to both face identification (1:N) and face verification (1:1). In addition, the standard uses the term face area, defined by four points (2.1: Bottom of the chin, 10.9: Upper contact point between left ear and face, 10.10: Upper contact point between right ear and face, 11.1: Middle border between hair and forehead) according to ISO/IEC 14496-2. The following section is based on [70].

Camera lens and related parameters

In order to avoid face distortion, it is crucial to set appropriate lens parameters and the distance between the face and the camera, which is called *camera to subject distance* (CSD) [70]. For example, in the case of an APS-C sensor with crop factor 1.44, we should consider using a lens of focal length between 50/1.44 and 130/1.44, or approximately 35 to 90 mm, while the CSD according to the best practices should be: 1.0-2.5 m (1:1) and 1.2 - 2.5 m. The camera must be placed at the subject's eye level with a tolerance of $\pm$ 5°. The objective of the metric is to control the magnification distortion that should ideally be less than 5 % (1:1) and less than 4 % (1:N), respectively. Similarly, the recommendation for the radial distortion is to maintain the properties under 2.5 %.

Focusing the camera from the nose to the ears should provide adequate sharpness. It follows that the recommended depth of field is approximately 15 cm from the level of the nose. The other important property to consider before taking a face photo is the *Inter Eye Distance* (IED). For legacy applications, the IED in the captured face photo must be greater than or equal to 90 pixels. However, for new passport applications, the IED should be at least 240 pixels. For a summary of the IED requirements, see Apendix A.

Scene background and illumination

A background surface needs to be positioned behind the head. In order to employ an appropriate background, it is essential for a background layer with a uniform color to be

placed to distinguish the facial components from the background. Although this standard requires the use of a uniform color, the use of a background with a gradient in one direction is also allowed. When considering the illumination of the scene, it is advisable to choose a uniform illumination with sufficient intensity. For automatic evaluation, this standard defines the four spots on the face where the mean intensity (MI) over all color channels is calculated. Then, within each spot and for each channel individually, the lowest MI should not be less than 50 % of the highest MI.

Contrast, dynamic range and color

To achieve appropriate face appearance, we need to choose suitable setup of the camera in terms of contrast, dynamic range and color, which have direct influence on image quality. In general, the parameters should be set to achieve sufficient contrast between face background and hair. Nowadays, the face images must be captured in color using cameras that allow capturing wavelengths between 400 nm and 700 nm, namely visible spectrum.

For color calibration, we should use the method with the 18 % gray background and the dynamic range of the image should have at least 50 % of intensity variation in the face region.

Head pose and face expressions

The rotation of the head is controlled by the cervical spine, but, in terms of this standard, the pose can be described by using three angles. To increase the suitability of a face image, we establish a goal to minimize these three angles. However, the standard allows small tolerances, which are defined by the following table, see Table 2.2.

Table 2.2: Pose angle requirements and recommendations [70].

Criterion:	Requirement	Pitch $\leq \pm 5°$, yaw $\leq \pm 5°$, roll $\leq \pm 8°$
Pose angle	Best practice	Pitch $\leq \pm 5°$, yaw $\leq \pm 5°$, roll $\leq \pm 5°$

We must not forget the last dynamic property of the face, the expression. In order to capture it, the face must have a neutral expression with the mouth closed. Likewise, the smile is not allowed even with the jaw closed. Furthermore, the eyes should open naturally, whereas the reflections on the iris should not exceed 15 % of the area. The standard defines *Eye Visibility Zone* (EVZ) [70] that should be visible and unobstructed.

Face accessories

In addition, this standard [70] deals with accessories that the person may wear for health and religious reasons. In short, sunglasses, glasses with polarization filters and tinted glasses are allowed, as well as glasses with a face image, provided that they are made of completely transparent material and are worn on the nose. Eyeglasses should not disturb the eyes or the EVZ. According to the reflectivity of the glasses, it is appropriate to adjust the light to reduce the number of light artifacts.

Portrait dimensions and head location

At the end of this section, we take a look at the requirements for portrait photography in terms of head size and localization. To ensure the fulfillment of the aforementioned

requirements, we put great emphasis on capturing the face accurately and appropriately, thus improving the overall quality. However, the last step is to define the set of ratios and parameters that are used to determine the location of a face in the image. We assume that a face in the image is surrounded by the specific border and is also centered, then it must meet the criteria in Table 2.3. The illustration is attached in Appendix A.

Table 2.3: Geometry requirements for portrait photo [70].

Term	Description	Requirement
A	Image width	-
B	Image height	$74\ \% \leq A/B \leq 80\ \%$
H	Line through the centers of the left (feature point 12.1) and the right eye (feature point 12.2)	-
M	Face center (midpoint of H)	-
M_h	Distance from the left side of the image to M	$45\ \% \leq M_h/A \leq 55\ \%$
M_v	Distance from the top of the image to M	$30\ \% \leq M_v/B \leq 50\ \%$
V	Line through mouth center (feature point 2.3) and M	-
w	Head width: Distance between the two imaginary lines parallel to the line; each imaginary line is drawn between the upper and lower lobes of each ear (feature points 10.2 / 10.6 for the right and 10.1/10.5 for the left ear	$50\ \% \leq W/A \leq 75\ \%$
L	Head length: Distance between the base of the chin (feature point 2.1) and the crown (feature point 11.4) measured on the imaginary line, if these feature points are not exactly located at 11.4, the vertical projection of them to shall be used	$60\ \% \leq L/B \leq 90\ \%$

Image format

According to this standard [70], several different image formats are defined for the representation of a face image, including the PNG, JPEG, JPEG2000, and RAW related to the camera. It is recommended to use lossless formats and it is crucial to capture the face image in color. In terms of image encoding, one of the following options should be used:

- JPEG Sequential Baseline (ISO/IEC 10918-1 [74])
- JPEG-2000 Part-1 Code Stream Format (ISO/IEC 15444-1 [75])
- JPEG-2000 Part-1 Code Stream Format (ISO/IEC 15444-1 [75])

- PNG (ISO/IEC 15948:2003 [76]) standard - PNG should not be used in its interlaced mode and not for images that have been JPEG compressed before

This section represented a brief summary of the ISO/IEC 19794-5 [70] standard. In addition to the information above, the document contains other information, such as recommendations for capturing children's faces, information related to the production of *Machine Readable Travel Documents* (MRTDs) [77]. It also includes data format requirements. The biometric passport is closely related to the standard because it is managed by the ICAO administration. Therefore, the face image of a passport must meet the requirements. Generally, a biometric passport contains a chip with biometric data and basic information about a person, such as first name, last name, birthday date, portrait photo and permanent address.

2.3.3 Performance metrics

For the evaluation of biometric systems and machine learning algorithms, it is essential to define standardized metrics [78]. In the field of biometrics, several metrics are used to evaluate a biometric system. First, we summarize the basic metrics related to detection and recognition success. The terms *genuine* and *impostor* express positive matches and negative matches, respectively.

Due to the varying imperfections of such a system, identity mislabeling can occur for both impostor and genuine pairs. Therefore, for biometric purposes, the essential metrics are derived from Figure 2.6.

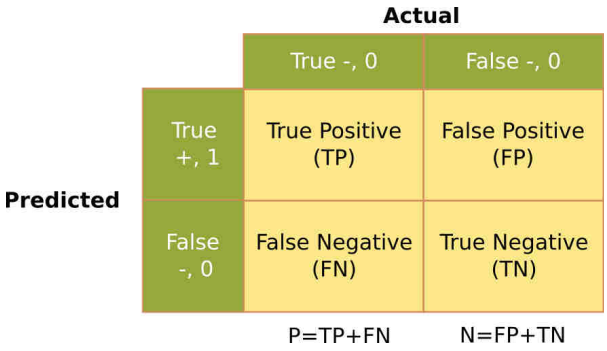


Figure 2.6: Systematic notations in a binary contingency table. Adopted from [79].

In the filed of machine learning, individual formulas are defined as follows [78]

- True positive rate (TPR) = Sensitivity = Recall = $\frac{TP}{TP+FN}$ (2.15)
- False positive rate (FPR) = $\frac{FP}{TN+FP}$ (2.16)
- True negative rate (TNR) = $\frac{TN}{TN+FP}$ (2.17)
- False negative rate (FNR) = $\frac{FN}{FN+TP}$ (2.18)
- Precision = $\frac{TP}{TP+FP}$ (2.19)
- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ (2.20)

where TP and TN denote correct positive matches and correct non-matches, respectively, and FP expresses instances where a positive match is misclassified. Finally, the negative FNs express the incorrect non-match when the match should be marked as positive.

However, in the ISO biometric standards, the names of the terms are different from those commonly used in machine learning. The following terms are related to the evaluation of biometric systems [TG.1]:

- False accept rate (FAR) = FPR
- False reject rate (FRR) = FNR
- True acceptance Rate (TAR) = TPR
- Equal error rate (EER) is error rate at which FAR and FRR are equal

.

The metric, known as the *Receiver Operating Characteristic* (ROC) curve [80], is used to express the relationship between FAR and FRR or also relationship between FAR and TAR [81], in order to determine the threshold of a biometric system. Theoretically, a perfect classifier would have a score placed in the top left corner of the ROC plot ($FPR = 0$ %, $TPR = 100$ %), while the worst classifier would have a score placed in the lower right corner. For a random classifier, the scores are typically distributed around the ROC diagonal. The point on the ROC curve where the FAR and FRR is equal is called equal error rate (EER) [82]. The threshold can be used to change and fine-tune the acceptance and rejection levels. Similarly, the threshold can be determined from the true and false distributions of the pairs.

Related to the ROC, there is another metric called the *Area Under Curve* (AUC) [83], i.e. for a single parameterized model that is defined as a single point. In the case of a parameterized model, the AUC is defined as the integral of the ROC function from (0,0) to (1,1). If the AUC value is zero, we consider the model prediction to be 100 % wrong, while the model prediction with $AUC = 1$ is considered to be 100 % correct. The AUC is desirable because it is scale invariant and classification threshold invariant. For illustration purposes, the ROC curve and the AUC metric are shown in Figure 2.7.

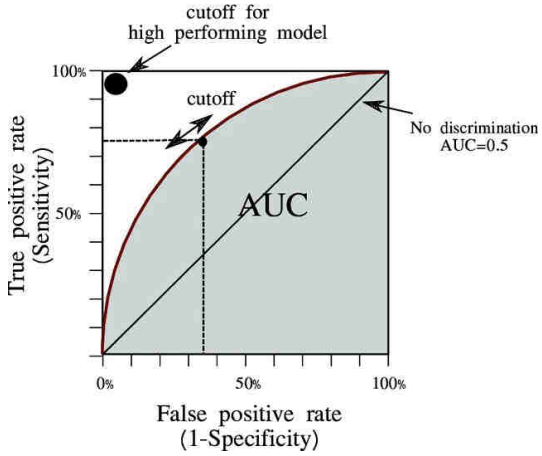


Figure 2.7: ROC curve with AUC [80].

The next metric close to ROC and AUC is *Average Precision* (AP), which is the average precision across all recall values between 0 and 1 [84] and also as area under precision-recall curve [85].

Another metric, the F1 measure, is the combination of recall and precision into a single score that is expressed by the following formula [78]

$$F1 = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \tag{2.21}$$

In addition to ROC, the histograms of genuine and impostor match scores can be used to analyze the performance of a biometric system, where the genuine score, also known as the similarity match score, expresses a correct positive match, whereas the term impostor score refers to the correct non-match between different subjects [86]. The distance can be represented by a matching score or by a distance metric with a range depending on its properties.

Chapter 3

Modern approaches for face recognition

With advances in computer technology, face recognition approaches have undergone significant changes over the years. Recently, the vast majority of face recognition systems are based on deep neural networks, see Figure 2.2. At first glance, face recognition can be seen as a multi-class classification problem, where each class corresponds to an identity (person). Related to this fact, the first approach based on a deep neural network to solve face recognition as a classification problem was published in 2014 [87].

The need to retrain the neural network model each time when a new identity is added is very restrictive. Another option is to create a descriptive vector (biometric template) for each face image, this process is called *face embedding*, see Section 3.5.1. Like face recognition, face detection is now also solved by a classification neural network.

Although the term neural network is familiar to almost everyone involved in computer vision, we would like to briefly summarize the basics of neural networks. However, the information provided here is mainly focused on neural networks for image classification or for solving regression tasks that process image data. We also introduce the necessity of face embedding and primarily discuss state-of-the-art approaches to face detection and recognition; see Section 3.4 and Section 3.5, respectively.

3.1 Theoretical background for neural networks

As an introduction, when we talk about neural networks implemented as a computer algorithm, we are talking about the so-called artificial ones that are inspired by *biological neurons* [88]. In general, the term neural network instead of artificial neural network is widely used in information technology.

An artificial neural network consists of an artificial system of neurons. The basic type of such systems is a feedforward neural network, usually represented as a three layer network [89]. The first layer is an input layer, which is responsible for the transfer of input data to the following layers. The output of this layer is connected to the subsequent parts of a model, including 0 to N hidden layers. Due to the goal of computer vision algorithms to get a response to an input, the last layer is called output layer. During the training process, the output of the output layer is compared to the expected output. From this comparison, the error is calculated using the *loss function* [88]. This evaluation shows how the neural network processes the input to produce the output. Another type of neural network

is the *recurrent neural network* (RNN) [90], where its connections between nodes can create a cycle in its graph.

So far we have only talked about neural networks. But what does the term *deep neural network* mean? How many layers must a deep neural network have? According to the Universal Approximation Theorem [91], any function can be approximated with a shallow network (three layer neural network). However, unlike deep neural networks, the number of neurons grows with the complexity of the task. Furthermore, in the book [92], it is stated that the number of hidden layers improves the accuracy. This fact was demonstrated by using a neural network model to transcribe multidigit numbers from photos. For these reasons, most current research is focused on the deep layer model.

Activation functions

To begin with, let us clarify what an *artificial neuron* [88] actually is. In this context, a neuron is basic unit, sometimes called a node, consisting of four entities: an input vector $x = x_1, ..., x_p$ along with weights $w = (w_1, ..., w_p)$, where p expresses the number of inputs to the neuron and a bias b and an activation function $g(x)$. Although the input of a neuron is exactly given by its initial input or the outputs of preceding neurons, the weights and biases are commonly altered during the training process. The basic principle is that the input vector is modified by the weights of the neurons and then partial outputs are summed. Such an output is called the net input and is defined as [88]

$$v_i = \sum_{j=1}^{p} w_{ij} x_j \tag{3.1}$$

where x_j represents the i^{th} element of the input vector and w_{ij} denotes the element of the weight vector.

The activation function $g(x)$ is the function that provides the response of the artificial neuron. In principle, the function maps the input of the network to the output of the nodes of the layer. Therefore, in mathematical notation, the artificial neuron is defined as

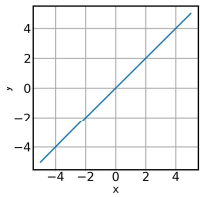
$$y_i = g(\sum_{j=1}^{p} w_{ij} x_j). \tag{3.2}$$

The choice of activation function has a significant effect on the performance of a neural network and its training process. We can divide the activation function into two categories, *linear* and *non-linear*. Let us look at common loss functions. The following summary is based on [93].

- *Linear* (Figure 3.1) - the first category of the loss functions is the linear activation function that preserves the input value.

In addition, there are non-linear activation functions that have been specifically designed during the development of neural networks.

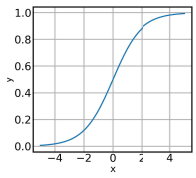
- *Sigmoid* (Figure 3.2) is monotonic function with a fixed output range (0,-1).
  Cons: The sensitivity of y depense on the change in x. The further the value of x is from zero, the smaller the value of the sensitivity is, which can lead to *vanishing gradients* [94]. The output of the loss is not zero-centered.



$$f(x) = y$$

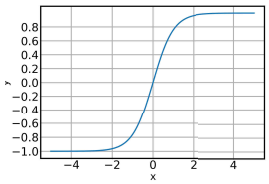
Figure 3.1: Linear activation function and its equation.

- Unlike Sigmoid, the *Tanh* (Figure 3.3) loss function is a zero-centered function.
  Cons: Because the derivatives are steeper, the gradient is stronger than the gradient of the sigmoid.
- *Rectified Linear Units (ReLU)* (Figure 3.4) is a modern activation function that is commonly used in the hidden layers. According to the book Deep Learning [92], the ReLU is recommended activation function for modern convolutional neural networks. The main advantage of this function is its resistance to the vanishing gradients problem.
  Cons: It should only be used inside hidden layers. Unfortunately, it can happen that the output of a neuron is zero for all data points. This is caused by the impossibility to adjust the weights during the descent. This phenomenon is called the dying ReLu problem [95].



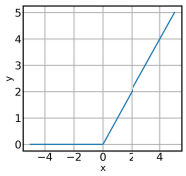
$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 3.2: Sigmoid



$$f(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Figure 3.3: Tanh



$$f(x) = \begin{cases} x & x > 0 \\ 0 & x <= 0 \end{cases}$$

Figure 3.4: ReLU

Figure 3.5: Non-linear activation functions and its equations.

- *Softmax* is a function that has a vector as an input. The vector has the same length as the number of classes, and its output is a probability distribution (the sum of all vector elements is one). The softmax loss function is utilized in classification tasks.

Each neuron in a neural network can be assigned a different activation function. In practice, however, all neurons in a layer are usually assigned the same activation function. Likewise, most hidden layers have the same activation function.

For the output layer, the activation function is chosen according to the solved task. For a summary of common output activation functions, see Table 3.1.

Table 3.1: Activation functions for output layers [96].

Task	Subtask	Activation Function
Classification	Binary	Sigmoid
	Multiclass	Softmax
	Multilabel	Sigmoid
Regression		Linear

Layers

Modern neural networks are composed of various types of layers, and do not have to be composed only of artificial neurons. From the perspective of deep neural network, the convolutional layer is considered to be one of the basic layers in convolutional neural networks.

In addition, modern neural networks require additional layers to achieve proper functionality and required performance. Such layers are used to extract information from an input and to improve the training process. For example, the additional layers are intended to ensure proper gradient descent. For example, the vanishing gradient problem can be suppressed with a certain composition of layers. The following text introduces most common layers used in a convolutional network.

The following layers are essential for creating the blocks used in a deep neural network.

- *Convolution layer* [97] - to explain what the convolution layer means, it is necessary to define the convolution operation. In the field of machine learning, the convolution for a discrete signal is used, which is mathematically defined as

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[g][n-m] = \sum_{m=-\infty}^{\infty} f[n-m]g[m]. \tag{3.3}$$

  The convolution is performed with the functions f and g, where the f expresses the signal that we want to process and where n is the step.

  However, for an image processing task, the expression has to be modified due to the above definition, which does not consider an image as input. Thus, a one-dimensional discrete signal in the equation has been replaced by the two-dimensional discrete signal, which is defined as $f[x, y]$. Furthermore, the two-dimensional filter $h[k_1, k_2]$ is used in the equations. Finally, the convolution is calculated as follows

$$(f \star g)[x, y] = \sum_{k_1=0}^{K_1-1} \sum_{k_2=0}^{K_2-1} h[k_1, k_2] \times [x + k_1, y + k_2]. \tag{3.4}$$

  Technically, the above expression is not a convolution, but it expresses a similar operation denoted as *cross-correlation* [97].

- *Pooling layer* [97] is often used to reduce the number of model parameters. The main idea is to map the response of input neurons within a given region to a single value. The value is calculated as the average of a neuron's outputs, the operation is called *average pooling* [97]. On the other hand, if the value is selected according to the maximum value in the given region, the layer is denoted as *max pooling* [97].

- *Fully connected* [97] is the type of layer that can be used to change the topology of the neural network. The outputs of all neurons in the previous layer are connected to all neurons in this layer. Therefore, each neuron is "fully connected" to all neurons in the input layer. If two fully connected layers are behind each other, the topology of the layer remains the same. In a regression task, the output of the fully connected layer can be interpreted as a vector of numbers, but in the case of classification tasks, the output of the neurons should be the probability of being a member of a class.

However, the layers described above are not the only ones used in convolutional neural networks. To develop a robust CNN model, it is necessary to use other types of layers, i.e. *add layer*, *concatenation* [98]. One of the advanced architectures is ResNet [98] uses add layer to merge the output of the convolutional blocks with the input. This technique is used to suppress the vanishing gradient problem [99].

Regularization of neural network model

Regularization [88][100] is a method to reduce the complexity of the model, as well as to suppress the bias and variance of the model. On one hand, the model can reduce its error by using the enormous number of assumptions, which of course implies a higher complexity of the model. On the other hand, reducing the variance of the model implies a simpler model. Therefore, for maximum performance, we need to balance these two aspects.

In order to mitigate overfitting and improve the generalization of a model, machine learning provides methods to reduce it [88]. With these methods, it is possible to reduce the variance while maintaining a similar bias. The first regularization methods, called L1 and L2 regularization [88], focus on reducing the range of neuron weights as close to zero as possible by adjusting the loss function. With the regularization methods, the weights of some neurons can be zero; therefore, the complexity of the neural network should be smaller. We should not forget about regularization methods, which we describe in more detail.

However, this is not the only way to ensure that the selected neurons are turned off. A special type of layer, *dropout* [100], can be used to randomly turn off neurons based on a set probability. According to the set probability, a certain number of neurons are turned off in each training cycle. As with the previous regularization method, the goal of regularization is to reduce the complexity of the model, resulting in a simplified network that is more resistant to overfitting.

Loss function

In neural network theory, the *loss function* [88] is intended to determine the error between the observed output and the predicted output of the algorithm (a model of neural network). In other words, the loss function attempts to quantify how close the predicted values produced by an algorithm are to the true values. Calculating the loss function is an important part of the neural network training process, and it allows optimization methods to determine the model parameters. Thus, the loss function can also be described as a training feedback signal.

The choice of a loss function depends on the nature of the task to be performed by the algorithm. In the following section, we present commonly used loss functions. We have chosen the loss functions primarily for the classification and regression tasks. The following selection of loss functions is related to classification tasks.

- *Cross entropy* [101] is a loss function used to solve multiclass classification tasks. It is also essential for other loss functions and is defined as follows

$$L_{CE} = -\sum_{i=1}^{n} y_i log(p_i) \tag{3.5}$$

where y_i expresses the truth label, n is the number of classes and p_i is the predicted probability for i^{th} class. This loss function assumes that the classified sample belongs to only one class.

- *Binary cross entropy* [102] is the modified version of the cross entropy loss which is determined for binary classification.

$$L_{BCE} = -\sum_{i=1}^{2} y_i log(p_i) = -y_1 log(p_1) + (1 - y_1) log(1 - p_1) \tag{3.6}$$

where y is the expected observation (truth label) and p is the predicted probability.

However, the selection of the loss function is also related to an activation function. When the binary cross entropy is combined with the sigmoid function, the neural network has the ability to solve the multiclass classification task where multiple labels can be assigned to the output. On the other hand, the binary cross entropy combined with softmax is intended to solve multiclass classification tasks; however, only one label is assigned to an output, see Figure 3.6, where C denotes the number of classes and t is the sample.

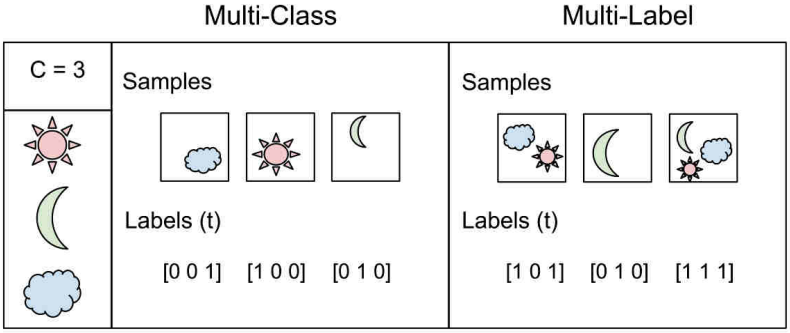


Figure 3.6: Difference between multi-class and multi-label classification [103].

Assuming that we want to use the probability distribution in output vector to classify the sample into one of the classes, the use of the softmax activation function and the cross entropy can be considered. The combination is also known as *softmax loss* and is derived as follows [104]

$$L_{CE_{softmax}} = -\sum_{i=1}^{C} y_i log(\frac{e^{z_i}}{\sum_j^C e^{z_j}}) \tag{3.7}$$

where z is the output vector and C expresses the number of classes. It is important to mention that some loss functions used in face recognition neural networks are derived from this loss function.

Let us look at the loss functions for regression problems [105]. The goal of these algorithms is to predict a real value. Therefore, the training datasets consist of samples and expected real numbers. For example, the samples can be images, and their expected values can be vectors containing the coordinates of key points in the image. Commonly used loss functions for regression tasks include [105]:

- Mean absolute (L1)

$$x = \frac{1}{N}\sum_{i}^{N} \|y_i - \hat{y}_i\| \tag{3.8}$$

- Mean square (L2)

$$x = \frac{1}{N}\sum_{i}^{N} (y_i - \hat{y}_i)^2 \tag{3.9}$$

where y_i is the ground truth value, $\hat{y}_i$ expresses the output of neural network and N is the number of samples.

Optimizer

The training of neural networks is ensured by the loss function and an algorithm for adjustment of attributes of neural networks called *optimizer* [106]. After each training step, the weights and other parameters in neural network are updated with the aim of reducing the error between observation and the expected output. Modern neural networks are made up of a large number of trainable parameters, and therefore it is almost necessary to use the algorithm that can help determine how to reduce the difference. In general, the aim of optimizer is to direct the adjustment of the weights of the model to reach the global minimum. In practice, we have to consider many aspects related to the neural network as well as training algorithms in order to get close to the local minimum.

- *Stochastic Gradient Descent* (SGD) [107] - due to the high computational complexity of the gradient within a batch, the SGD was proposed. The algorithm selects only one sample from the batch to calculate the exact value of the gradient. The computational cost of the algorithm is independent of the number of samples. In the case of strongly convex problem, the SDG can achieve the optimal convergence speed.
- *Adam* [107] - improves the SGD by utilizing an adaptive learning rate for each parameter. To ensure the gradient descent, the algorithm combines adaptive learning rate and momentum methods.

3.2 External datasets for face detection and recognition

In machine learning, data are an irreplaceable part of the training, testing, and validation process. Such a collection of data is commonly organized into a set denoted as a dataset. Generally, the aim of machine learning (ML) algorithms is to extract information and provide a way to process it. In this section, we focus on the datasets utilized by algorithms related to face recognition algorithms.

As we already know from the Eigenface and Fisherface algorithms, the classification problem can be solved using supervised or unsupervised learning. In the case of unsupervised learning, a machine learning algorithms utilize a data without label, and therefore

the algorithm does not consider the assigned response of the data sample. Supervised learning, on the other hand, checks the response of an input sample against a reference label to determine whether or not it was correctly classified during training.

Duing the training process, the classification parameters are adjusted at each iteration to minimize the error of the task. Similarly, regression tasks can be solved with ML algorithms trained with a supervisor.

Let us examine the partitioning of the datasets. Remember that the datasets are used to test and validate machine learning algorithms. To evaluate performance, the trained model is tested on another dataset called *validation dataset* [108]. Unlike the training dataset, which is used directly to change the parameters of the neural network model, the validation dataset can be used to adjust the hyperparameters if the model's performance is not suitable.

The final type of a dataset is *test dataset* [108], which is designed to evaluate trained algorithm or neural network model. Unlike the validation dataset, the test dataset has no influence on the training process. It is only used to evaluate the performance of the learned algorithms.

When creating the dataset, we need to determine how many samples will be dedicated to the training, validation, and test datasets. While [109] provides recommendations for partitioning the dataset, we must also consider the specific task at hand and adjust the partitioning.

Because face recognition algorithms use datasets composed of images, the following information applies primarily to this type of dataset. However, the number of samples in a dataset must be considered in almost all machine learning tasks. Since the size of the dataset has a significant impact on the final performance of the model, we must take this into account when designing and training a neural network. Furthermore, one of the crucial properties of a dataset is the balance of samples across classes [110], which can directly influence the performance of algorithms across classes.

Satisfying the requirements when creating a dataset can be time consuming and expensive. Therefore, the data in the dataset can be modified to obtain a larger dataset consisting of data represented in different situations. This can be accomplished by a method called *augmentation* [111] and its idea is based on using image processing algorithms to modify the existing samples and thus extend the dataset. The former has the advantage of faster training, while the latter has the ability to generate unique samples for each training iteration. Among the well-known methods for image modification are *cropping*, *rotation*, *flipping*, and *translation* [111]. However, other computer vision operations can also be used. Furthermore, augmentation has been shown to reduce overfitting [111].

It is important to note that the data in the dataset should represent the real situation in which the model will be used. In the case of face images, we can find the terms *in the wild* or *unconstrained conditions* [26] to express that the data were captured outside of controlled conditions. This means that the face images were captured regardless of the position of the face, lighting conditions, etc. On the other hand, if a face image is captured with the direct goal of capturing it, then we speak of a controlled capture.

Another important factor that can affect the performance of the trained algorithm is the alignment of object of interest [112]. For example, the output of face detection algorithms may be coordinates that mark the position of the face. However, the center position of the face in the region is not guaranteed. If the face is misaligned in the region, the trained face recognition neural network may perform poorly.

In the following section, we present three main categories of datasets that can be utilized by face recognition algorithms. It is necessary to state that the presented selection of datasets is influenced by our opinion and in practice, it is difficult to objectively assess the significance of individual datasets. Moreover, some datasets are intended to solve more than one task, for example, the AFW [113] contains samples for both face recognition and landmark regression.

3.2.1 Face detection

In the first section, we focus on datasets that are utilized by face detection algorithms. Historically, after the introduction of the Viola-Jones and the Histogram of Oriented Gradients face detector (Section 2.2.1), the researchers began to develop new possibilities for face detection. The main need for datasets came with the advent of solutions based on neural networks. Nowadays, the datasets are composed of face images ranging from easy, controlled capturing scenarios, to very difficult in-the-wild scenarios.

FDDB

The dataset called *Face Detection Data Set and Benchmark* (FDDB) [114] was introduced in 2010 and is presented as a benchmark for testing the face detection algorithms in unconstrained scenarios. An interesting feature of the dataset is that faces are labeled with ellipses. In contrast to the face detection dataset, where faces are usually labeled with rectangles, this brings new possibilities for the accurate evaluation of face detection. The number of faces in the dataset is 5,171 faces in 2,845 images.

Year: 2010
Metric: 5,171 annotated faces

AFW

The *Annotated Faces in the Wild* (AFW) dataset was introduced in [113] and is intended for testing algorithms. It is a relatively small dataset consisting of 468 faces in 205 images collected from Flickr[1] images. The selection was made with an emphasis on covering a wide variation in both face viewpoint and face appearance. Each face was tagged with a bounding box, a discretized viewpoint, and at most six significant landmarks.

Year: 2012
Metric: 468 annotated faces

MALF

With the continuous improvement of face detectors, new requirements for datasets have emerged. In particular, the dataset is being expanded to include images taken under uncontrolled conditions. This was taken into account by the authors of the *Multi-Attribute Labeled Faces* (MALF) dataset [115]. In addition, each face has meta-information associated with it, including, in addition to bounding box and pose, facial attributes such as gender, glasses, occlusion, and exaggerated expression. The dataset is primarily intended for a fine-grained evaluation of face detection, while it consists of 5,250 images with 11,931 annotated faces collected from the internet.

[1] www.flicker.com

Year: 2015
Metric: 5,171 annotated faces

WIDER Face

In 2017, a further significant dataset for the training and evaluation of face detection algorithms was introduced, significantly exceeding the number of images available at that time [116]. The dataset contains 32,203 images with 393,703 labeled faces. Moreover, the authors demonstrate limitations of selected face detection algorithms available at the time, such as heavy occlusion, small scale, and atypical pose. The selection of images for the dataset emphasizes variations in scale, occlusion, pose, and background clutter.

This data is divided into ten classes, determined by events defined using the *Large Scale Ontology for Multimedia* (LSCOM) [117]. For each category, the 1,000-3,000 matching images were found using search engines such as Google and Bing. In the last step, the data were processed by manuall examination of all the images and also filtering the images without faces.

Finally, the authors divided the dataset into training, validation, and testing, such that 40 % is intended for training, 10 % for validation, and 50 % for testing. According to the difficulty of pose and occlusion of the faces in the images, the dataset is further divided into three categories: easy, medium, and hard.

Although the dataset is popular, perhaps the most popular one, the paper lacks any information regarding the license agreements for individual images.

Year: 2017
Metric: 393,703 annotated faces

3.2.2 Facial landmarks

Among the algorithms related to face analysis and recognition are the algorithms for determining the positions of facial landmarks. The obtained landmarks can be used by various applications such as face alignment algorithms, facial expression detection, face deepfake generation, and more. The challenge arises from the need to accurately locate the points, and this challenge is multiplied by the number of points to be detected. Moreover, some facial landmarks that can be detected by available approaches intersect with anthropometric points. In general, the accuracy of the position of such landmarks is critical for applications that use landmarks for anthropometric purposes. On the other hand, for many tasks, a small inaccuracy of facial landmarks does not pose a problem. The positional accuracy of landmarks is generally better for well-defined points, such as the tip of the nose, the corner of the eyes, and the corner of the mouth.

AFLW

The *Annotated Facial Landmarks in the Wild* (AFLW) [118] dataset introduced in 2011 is another representative of datasets composed of images with a wide variety of appearances. It consists of 25,993 real images gathered from Flicker. The sample labels were chosen to primarily mark the internal keypoints of the face, rather than the points on the outline of the face. The number of labeled points per face image varies according to their visibility with the assumption that the 21 points are the maximum that can be determined. The AFLW dataset is well suited for both training and testing.

COFW

The *Caltech Occluded Faces in the Wild* (COFW) [119], presented in 2013, was created with the participation of four annotators with different levels of computer vision expertise. Each annotator collected 250 images to represent real world situations. The images were then manually annotated with the same landmarks as in the LFPW dataset. Consequently, the subset of images was annotated twice to measure human performance. The final dataset consists of 1,007 images with 29 landmarks.

300 Faces In-The-Wild Challenge

The next representative of annotated face datasets is *300 Faces In-The-Wild Challenge* (300W) [120], which was an essential part of the facial landmarks challenge in 2013 and 2015. The dataset is divided into two parts, where the first part is represented by faces captured in indoor environments and the second part includes images captured in outdoor environments. Each part contains 300 faces with 68 labeled landmarks.

WFLW - Wider Facial Landmarks in-the-wild

One of the datasets that contains images taken in the wild is the *Wider Facial Landmarks in-the-wild* (WFLW) [121] dataset. This dataset consists of faces with a large variation in expression, occlusion, and pose, making it possible to assume that the images were taken in the wild. Unlike the other datasets presented, the WFLW dataset includes 10,000 faces, each annotated with 98 landmarks. Overall, the dataset is suitable for training robust face landmark detectors.

3.2.3 Face recognition

In the following section, we describe the collection of widely-used datasets that are used to train and evaluate face recognition algorithms.

Labeled Face in the Wild

One of the most important benchmark datasets still used today to evaluate face recognition algorithms is the dataset called *Labeled Faces in the Wild* (LFW) [122], which was published in 2007. The dataset consists of 13,233 face images of 5,749 individuals collected from the web. The number of samples per person varies throughout the dataset, and only 1,680 identities have two or more images. The images were detected using the Viola-Jones detector, which is now considered obsolete.

The LFW dataset is intended for limited academic purposes due to its inherent limitations that could potentially bias the results. One of these limitations is the underrepresentation of certain groups in the dataset. For example, children and individuals over the age of 80 are minimally represented, and babies are not included at all. Furthermore, the ethnic representation in this dataset is not balanced, and some ethnicities may be missing entirely. The distribution of men and women across the dataset is also unbalanced. In [87] is stated that the LFW contains about 75 % males.

The next drawback is the statistically significant number of samples among subgroups. This means that the number of samples in the data is statistically inappropriate for evaluating algorithms within and between subgroups. Another limitation stems from the method of image capture; the images in the dataset are of good quality, close to studio photographs,

because the dataset consists mainly of celebrity photos, and the images were mostly taken by professional photographers. The images taken under difficult conditions, such as extreme pose, strong occlusion, and others, do not make up a large part of the LFW.

According to its authors, the dataset is primarily intended for the research purposes. Therefore, the LFW disclaimer states that the dataset should not be used to evaluate commercial algorithms.

Currently, there is one original dataset and three additional datasets that differ in the alignment methods used. The first additional dataset contains "funnaled images" (2007) and the *LFW-a* dataset (2012) consists of images aligned by an unknown method. Images aligned by deep funneling are included in the last dataset.

In addition, the *Cross-Age LFW* (CALFW) [123] dataset was introduced in 2017 [123] to suppress the age imbalance drawbacks of LFW, see Figure 3.7.

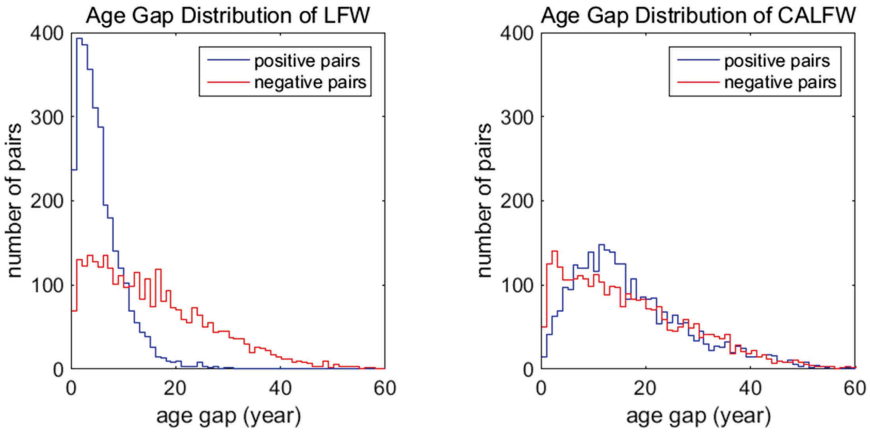


Figure 3.7: Age gap comparison [123].

As a result, the LFW is still a popular benchmark for evaluating face recognition algorithms. In Section 3.5, we present the face recognition neural networks that have been evaluated for their performance on the LFW.

YouTube Faces Database

The *YouTube Faces Database* [124] is a dataset intended to study the problem of unconstrained face recognition in videos. The dataset is made up of 3,425 videos from 1,595 different subjects. Each subject in the dataset is represented in one to six videos. The process of creating the dataset involved searching for videos on YouTube using the subjects' LFW labels. Six videos for each subject were then selected and downloaded and these were then divided into frames at a rate of 24 frames per second.

Similar to LFW, the faces were detected using the Viola-Jones detector. The resulting faces were then filtered to suppress detections that occurred for less than 48 consecutive frames. The term consecutive frames was defined as detections with an Euclidean distance between their centers of less than 10 pixels. Furthermore, all regions of interest (ROIs) of the detections are scaled by a factor of 2.2 of their original size and cropped from the frame. These cropped images have a size of 200×200 pixels, which are subsequently scaled down to 100×100 pixels.

As a test benchmark, the authors introduced a list of pairs. The benchmark is based on randomly collected pairs, half of which are equal matches and the other half of which are mismatches. The pairs are then divided into 10 mutually exclusive subgroups. Overall, the dataset partially complements the LFW dataset and extends it with the face images captured in an uncontrolled environment.

AgeDB

In 2017, another dataset, *AgeDB* [125], was introduced for the verification of the face recognition algorithms. The main characteristic of the dataset lies in the wide age distribution of the subjects. The subjects are manually annotated, so the author considers the dataset to be noise-free. In addition to the identity labels, the images are also annotated with gender and age at the time of the dataset release.

In total, the dataset contains 16,488 images of 568 different subjects. The ages range from 1 to 101 years, with an average age of 50.3 years.

MS-CELEB-1M

One of the largest datasets for face recognition was introduced by Microsoft in 2016 [126]. The dataset contains approximately one million images of 100,000 individuals. Although the name of the dataset includes the word "celeb", the images in the dataset belong to individuals who need to maintain an online presence for their professional lives such as journalists, artists, musicians, activists, politicians, writers, and academics. The authors of the dataset have prepared scraping algorithms that can assign meta information in addition to the individual's name.

The authors used a knowledge graph called Freebase [127] to define one million identities. The first step was to select a subset of identities, ensuring that each identity had a positively defined facial appearance. The items in the subset were then ranked based on their frequency of occurrence on the web. Finally, the top one million items from the ranking were selected.

This approach was adopted for two main reasons. The first reason is that the authors focused the benchmark on a real-world application. The second reason is that the availability of real images for identities increases the likelihood that the images belong to the correct identity.

Due to the fact that the dataset is intended for training and testing face recognition algorithms, the part of the dataset intended for testing is separated. In the published dataset, 1,500 celebrities were selected for measurement, each with two face images. Since the authors emphasized accuracy, the images for each celebrity in the subset were manually reviewed.

In the paper, the capabilities of the dataset were demonstrated on 100,000 identities, with a selection of 100 images per identity. The main goal of the paper was to find an effective approach for scraping data from the internet.

This dataset was introduced around 2016. Nowadays, the presented approach is controversial. The Microsoft website with the dataset was closed in 2019. In [128] it was presented that there is a suspicion that this dataset could have been used to track a foreign journalist in China.

Although the dataset website has been discontinued, the dataset is still available on the internet. Moreover, it has also been used in published papers. Some variants of this dataset have been published nowadays, i.e. C-MS-Celeb [129]. The difference between the original dataset and the "forked" dataset is that the mislabeled images have been removed. The

assumption is that the performance of the trained neural network will be better with this dataset.

CASIA-WebFace

CASIA-WebFace [130] is another dataset for training a classifier or embedding a neural network for face recognition, consisting of images that were scraped from the internet. The dataset was created by Chinese Academy of Sciences[2] and is available for non-commercial and academic purposes. The dataset contains 94,414 face images of 10,575 real identities. From 3/2023, the dataset webpage is unavailable.

DigiFace-1M

Due to restrictions on individual privacy, datasets containing real faces can raise legal and moral concerns. This leads to new approaches to collecting new datasets. The authors of the *DigiFace-1M* [131] dataset established a goal to address three common shortcomings. The first stems from the ethical issues raised by the fact that the face images in the dataset are used without explicit consent. Second, a face dataset may contain mislabeled images, which affects the performance of an algorithm. By using the generated dataset, the correctness of the labels is guaranteed. The last problem that often occurs in a real face dataset is data bias, which includes unbalanced racial distribution and celebrity photos with specific appearances that lead to inadequate representation of individuals.

Despite overcoming these shortcomings, face images may exhibit non-standard face features that deviate from anthropometric characteristics, see Figure 3.8. Our approach [TG.2] to generating the dataset has a better assumption that the face features are preserved.

The dataset introduced by Microsoft in 2023 is divided into two parts, where the first part consists of 720,000 images with 10,000 identities and the second part consists of 500,000 images with 100,000 identities.

Figure 3.8: Face images from DigiFace-1M dataset (the face images are available in low resolution only) [131].

[2] http://www.nlpr.ia.ac.cn/en/

3.2.4 Family of IJB datasets

Among the important datasets intended for face detection and recognition evaluation, we considered the family of *IARPA Janus Benchmark* (IJB) datasets [132][133][134]. In contrast to previous datasets, these datasets were introduced by the prominent institutes NIST[3] and IARPA[4], which perform an independent[5] evaluation of algorithms. Public challenges began with the launch of IJB-A in 2015 [132].

Let us look closer at the dataset. The crucial features of the dataset include full pose variation, possibility to use a dataset for face detection and face recognition, samples consist of images and videos, wider geographic variation of the subjects, protocols for identification and verification, ground truth of eye and nose location and so that the optional protocol allows modeling of gallery subject. And all this for 500 subjects in 5,712 images under Creative Commons license[6].

The next generation of the dataset was introduced in 2017 and is referred to as IJB-B [133]. In contrast to the previous dataset, the number of subjects increased to 1,845, where 21,798 images and 7,011 videos were annotated. At the time of publication, it was the largest annotated unconstrained joint detection, recognition, and clustering benchmark. The dataset covers 10 different protocols to satisfy the mentioned domains. The IJB-B protocols and the results of Government-Off-The-Shelf (GOTS) [135] algorithms have been evaluated and published.

The last generation of IJB is represented by the IJB-C dataset [134]. Similar to the previous dataset, the number of subjects has been further increased to 3,531 within 31,334 images and 11,779 videos. The majority of the dataset consists of face images of celebrities, such as actors and performers, which are strongly associated with physical appearance and may be less representative of the global population. To avoid these drawbacks, the authors used other sources to ensure the diversity of the subjects' professions.

3.2.5 Datasets with face in varying angels

Multi-PIE [136] - other datasets include the Multi-PIE, which consists of images from 337 subjects taken in four sessions over six months, and includes faces taken from exact viewpoints. This dataset is no longer available.

Celebrities in Frontal-Profile in the Wild (CFP) [137] - the dataset consists of frontal and nonfrontal images of celebrities, which is its strength over others. The dataset contains images of 500 individuals with 10 frontal images and four non-frontal images. To evaluate the algorithms, the scenarios were prepared for both frontal-frontal and frontal-profile comparisons.

Headpose [138] - although the dataset consists of only 15 individuals, there is an advantage in the number of images taken at specific poses.

UMDFaces [139] dataset - the next interesting representative of face datasets is UMDFaces that contains 367,888 face annotations for 8,277 subjects. Each face is annotated with yaw, pitch, and roll. Unfortunately, the dataset is temporarily unavailable.

[3] National Institute of Standards and Technology
[4] Intelligence Advanced Research Projects Activity
[5] Some companies do not consider this testing to be independent.
[6] https://creativecommons.org/licenses/

3.3 Deep features extractor - backbone

In modern convolutional neural networks, the stage responsible for feature extraction is referred to as *backbone* [140]. This is an essential part of the neural network that is often trained on a different task, which also demonstrates its capabilities [140]. The known backbones are primarily trained as image classification or object detection tasks. For the use in other types of tasks, the last fully connected layer is removed and replaced by the head, thus creating a new model.

Using one of the known backbones, we can save time in designing a neural network. Of course, we can propose our own backbone, but improving the capabilities of a feature extractor is difficult because there are many parameters to be considered. Overall, most face detectors and recognition models are based on neural networks using one of the backbones presented below.

However, a detailed description of the backbones is beyond the scope of this thesis. Therefore, we present only a simplified summary of the backbones to highlight their capabilities to be used in embedded systems to solve face detection and recognition tasks.

The pioneer of well-known backbones is *AlexNet* [141], introduced in 2012 for image classification, which consists of convolutional, max-pooling, and dropout layers. In total, AlexNet contains 60 million parameters and 650,000 neurons. AlexNet has been succeeded by families of backbones, such as *VGG-Net* [142], *ResNet* [98], *Inception* [143] and *DenseNet* [144].

Inception backbone is composed of Inception modules [140]. This module is composed of convolutional blocks of different kernel sizes, with the layers arranged in parallel. This means that the input of the Inception layer is broadcast to the layers and their output is merged into one, see Figures 3.9 and 3.10.

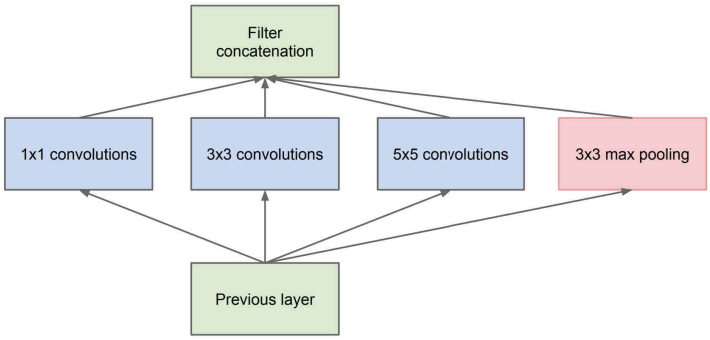


Figure 3.9: Inception module [145].

Among the new generation of backbones is EfficientNet, which was released in 2019 [146]. In general, improving the performance of neural networks can be achieved by scaling the architecture. Note that known architectures include subtypes according to depth. For example, the ResNet architecture can be scaled by the number of convolution blocks from ResNet50 to ResNet300.

Next possibility to change the number of parameters is to scale a neural network architecture up by the width by changing the resolution of an input image. Given the assumption that neural network performance depends on depth, we can change the neural network topology to achieve the required performance for the specific task.

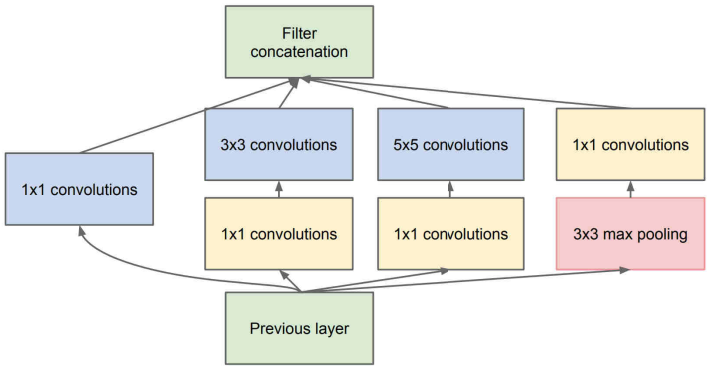


Figure 3.10: Inception module with dimension reductions [145].

The first empirical quantification of the relationship among the three dimensions of the width, depth and resolution of the network was presented in [146]. The authors focused on balancing the complexity, number of parameters and performance of the neural network with the goal of creating a new generation of backbones. As part of the research, the authors designed new backbone architectures called EfficientNet. The cornerstone of the architecture is the inverted bottleneck *MBConv* [147].

To select relevant blocks and their parameters, the design of the architecture was conceived as an optimization task to optimize $ACC(m) \times [FLOPS(m)/T]$, where $ACC(m)$ and $FLOP(m)$ express the accuracy and FLOPS of the model m, respectively. The parameter w is used to control the trade-off between accuracy and FLOPS. Furthermore, the other architectures of the EfficientNet family were derived from the baseline using the scaling method, which determines appropriate parameters to describe the depth, width and resolution of the base model. The parameters are then locked and scaled by the scaling factor.

When it was first released, EfficientNet outperformed most of the backbones available at the time [146]. Surprisingly, according to our research, EfficientNet has not yet been thoroughly explored in terms of face recognition. Nowadays, the next generation of EfficientNet has been introduced, called EfficientNetV2.

Although the accuracy of these backbones has improved over time, the complexity of their architecture has also increased. Therefore, the use of such backbones in an embedded solution is limited. The need to use neural networks in mobile and embedded applications led to the development of backbones that focused on processing speed performance. The first significant attempt to meet these requirements was made by Google with the MobileNet backbone. The essential part of the backbone is the use of depth-wise convolutional layers to reduce the computational cost (similar layers are used in Inception). There are currently many versions of MobileNet [148][149].

If we decide to use a platform to accelerate a neural network, we need to know what operations the platform supports. In the case of EfficientNet-like backbones, the base of the architecture has been modified to a lightweight version for effective computation by Google TPU [150]. The main goal of this version of EfficientNet is to ensure effective computation on TPU. In addition, EfficientNet-Lite has a similar accuracy on ImageNet with lower latency than Resnet50, see Figures 3.11 and 3.12. Likewise, the MobilenetV2/V3 is also intended for embedded and mobile devices.

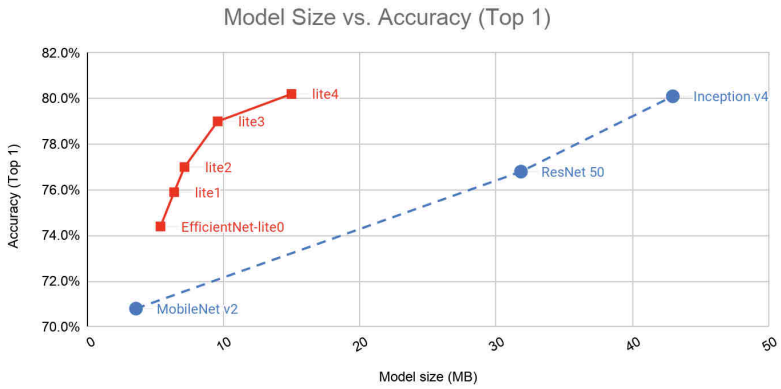


Figure 3.11: Latency comparison of EfficientNet-Lite to MobileNetV2, ResNet50 and Inception v4 [151].

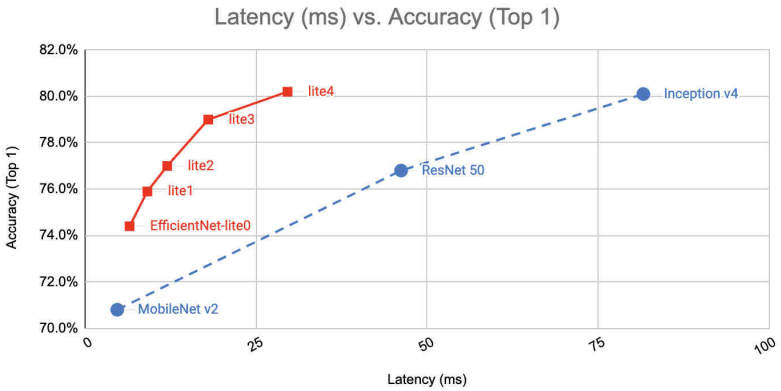


Figure 3.12: Model size comparison of EfficientNet-Lite to MobileNetV2, ResNet50 and Inception v4 [151].

3.4 Face detection

Face detection is the first stage in the face recognition process, where the goal is to precisely locate the face area in an image while minimizing unnecessary background inclusion. Since the face detection task can be safely replaced by general object detection, in the following sections we discuss the concept of object detection in general.

Historically, face recognition has used a technique known as the sliding window method, usually in connection with Histogram of Oriented Gradients (Section 2.2.1) and classification algorithms such as Viola-Jones (Section 2.2.1). As the name implies, this method involves a window that systematically traverses the image in predefined increments. A binary classification is typically performed at each position. However, this approach has several limitations – for one, it assumes that the object to be detected exactly matches the dimensions of the sliding window, which proves restrictive in real-world scenarios where objects of the same type can vary significantly in size. This limitation also extends to the aspect ratio of the sliding window. Another challenge is determining the appropriate step size for

window movement. A larger step can result in missed object detection, while a smaller step significantly increases the number of classification operations.

Due to the inefficiency of the techniques, the performance of detection in real time is limited. This led to the replacement of the method in modern algorithms. Around 2013, approaches using neural networks began to be published, and the sliding window over randomly generated windows changed to one-stage detectors with the ability to locate objects in an image.

Let us look at two concepts commonly used in object detection algorithms. To evaluate the positional accuracy between the detected bounding box and its reference, the metric *Intersection over Union* (IoU) [152] is commonly utilized. The metric expresses the intersections between the predicted bounding boxes and their ground truth bounding boxes. Mathematically, the metric is defined as [152]

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{3.10}$$

where $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$ and $\mathbb{S}$ expresses the shape of the detection.

If $|A \cap B| = 0$ then $IoU(A, B) = 0$. Thus, IoU does not reflect the positional relation of the shapes, which is considered the main drawback [152] of the metric.

We also focused on *Non-Maximum Suppression* [153] (NMS), which is an algorithm for filtering a set of proposed boxes with the goal of finding the one that best fits the object. The algorithm computes the IoU metric to express the relationship among the boxes.

The algorithm is divided into several steps, where the input contains proposed boxes B, corresponding confidence scores S, and overlap threshold IoU N. The expected output is a list of filtered prediction bounding boxes. The algorithm follows the steps described in Algorithm 1.

Algorithm 1 Non-maximum Suppression algorithm [153].

Input:
$\mathcal{B} = \{b_1, ..., b_N\}$; $\mathcal{B}$ - is the list of initial detection boxes
$\mathcal{S} = \{s_1, ..., s_N\}$; $\mathcal{S}$ - contains corresponding detection scores
N_t - is the NMS threshold
$\mathcal{D} \leftarrow \{\}$
while $\mathcal{B} \neq empty$ **do**
  $m \leftarrow argmax(\mathcal{S})$
  $\mathcal{M} \leftarrow b_m$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
  for $b_i in \mathcal{B}$ **do**
    $iou\_m \leftarrow iou(M, b_i)$
    if $iou\_m >= N$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
    end if
  end for
  return $\mathcal{D}, \mathcal{S}$
end while

The basis of the algorithm is parameterized by the threshold. Therefore, in certain cases, the correct detection may be suppressed. We assume three object proposals that overlap

with a confidence score around 0.8. Although these are three different objects, in the second step, the correct object can be eliminated when IoU is higher than the set threshold. This drawback has a significant impact on the accuracy of the detection. Fortunately, the algorithm has been modified and the drawback has been suppressed. The improved version of the algorithm is called Soft-NMS and it is described in Algorithm 2 [153].

Algorithm 2 Soft Non-maximum Suppression algorithm [153]

Input:
$\mathcal{B} = \{b_1, ..., b_N\}$; $\mathcal{B}$ - is the list of initial detection boxes
$\mathcal{S} = \{s_1, ..., s_N\}$; $\mathcal{S}$ - contains corresponding detection scores
N_t - is the NMS threshold
$\mathcal{D} \leftarrow \{\}$
while $\mathcal{B} \neq empty$ **do**
  $m \leftarrow argmax(\mathcal{S})$
  $\mathcal{M} \leftarrow b_m$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
  for $b_i \in \mathcal{B}$ **do**
    $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$
  end for
  return $\mathcal{D}, \mathcal{S}$
end while

where the $s_i f$ is defined by

$$s_i = \begin{cases} s_i, & \text{iou}\left(\mathcal{M}, b_i\right) < N_t \\ s_i \left(1 - \text{iou}\left(\mathcal{M}, b_i\right)\right), & \text{iou}\left(\mathcal{M}, b_i\right) \geq N_t \end{cases}, \tag{3.11}$$

Modern object detectors can be categorized by the number of stages. The most commonly used object detectors consist of one or two stages. However, the detectors based on the single-stage concept are considered the state-of-the-art detectors [154][155].

3.4.1 R-CNN

The *Region with CNN Features* (R-CNN) [156] algorithm was published in 2013 and is one of the first approaches using CNN to localize and classify objects in an image. This detector system consists of three modules, see Figure 3.13. The first is intended to generate category-independent region proposals. Similar to the sliding window, the object classification is performed for each proposal. Thus, in order to fulfill the detector function, there are two other modules to ensure object classification. Since the CNN part of the system has a fixed input size (resolution 227×227 was used in the paper), the proposed regions have to be adapted to the size. In the next step, each object proposal is mapped to a 4096-dimensional feature vector and then classified by the SVM algorithm (Section 2.2.1). The detector may generate more regions for individual objects, which should be reduced by the post-processing algorithm. One of the ways to reduce the proposals brings the NMS algorithm, which utilizes the IoU score of the covered bounding boxes for each class independently.

The speed capability of the algorithm is presented in [156]. The test scenario described by the authors in this paper uses about 2,000 proposals generated by the Selective search algorithm. Computing region proposals and features took 13 seconds on GPU and 53 seconds on CPU (the CPU and GPU specifications are stated in [156]). Nevertheless, the speed of

the detector system is one of its main drawbacks. On the other hand, it is possible to achieve high accuracy of object detection with this detector.

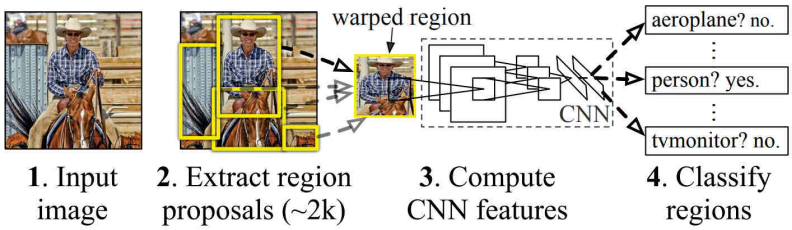


Figure 3.13: Architecture of R-CNN object detector [156].

3.4.2 Fast R-CNN

The next iteration of R-CNN brings two algorithms, the first being Fast R-CNN (2015) [157]. Like R-CNN, the Fast R-CNN uses *Selective search* [158] to generate proposal regions. In contrast to R-CNN, the architecture of Fast R-CNN combines the proposed regions with the feature map generated by the CNN neural network. For feature extraction, one of the possible options is to use a well-known backbone (see Section 3.3), or it is also possible to design your own architecture to perform feature extraction. Furthermore, each extracted feature vector is fed into a sequence of fully connected layers, which are divided into two branches. The first branch is responsible for predicting the probabilities of K object classes using softmax. The output of the second branch is composed of four real values representing the position of the bounding box, which is labeled by one of the K object classes. Moreover, like R-CNN, the regions proposed by Fast R-CNN have varying sizes, thus they need to be unified. The authors designed and implemented a layer called ROI pooling, which aims to map regions of different sizes to regions of a uniform size. The ROI pooling layer is based on the spatial pyramid layer with only one pyramid level. The input layer is divided into subwindows of size h/H by w/W, where h and w denote height and width, and H and W express the required number of subwindows. By pooling each sub-window, we obtain a fixed size of output features, which are then fed into fully connected layers. The architecture is shown in Figure 3.14.

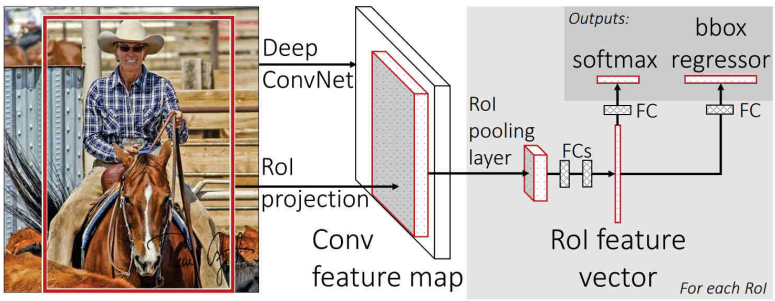


Figure 3.14: Architecture of Fast R-CNN object detector [157].

Due to the two different outputs of the neural network, a loss function should be utilized to process both. The first of them is the classification vector $p = (p_0, \ldots, p_K)$, which is represented by a discrete probability distribution (per ROI), where K is the number of cate-

gories. In the case of the bounding box regression, the output consists of $t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$ for each of the K object classes. Thus, the loss function is defined as [157]

$$L\left(p, u, t^u, v\right) = L_{\mathrm{cls}}(p, u) + \lambda[u \geq 1] L_{\mathrm{loc}}\left(t^u, v\right) \tag{3.12}$$

where u,v are ground truth class and ground truth bounding box, respectively, L_{loc} expresses the ground truth bounding box and predicted tuple for class u in the vector t, $L_{\mathrm{cls}}(p, u) = -\log p_u$ is log loss function for true class u defined by following loss [157]

$$L_{\mathrm{loc}}\left(t^u, v\right) = \sum_{i \in \{\mathrm{x,y,w,h}\}} \mathrm{smooth}_{L_1}\left(t^u_i - v_i\right), \tag{3.13}$$

where $smooth_{L_1}$ [157] is defined as follows

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3.14}$$

Fast R-CNN achieved state-of-the-art performance on VOC2012[7], VOC2010[8] and VOC2007[9] when it was released.

3.4.3 Faster R-CNN

Another type of R-CNN is the *Faster R-CNN* algorithm [159], which, to put it bluntly, uses only CNN without any other supporting algorithms. Unlike other R-CNN approaches, the proposal windows are generated directly by the convolutional neural network. The high-level idea is to feed an image into a neural network and use its responses, including bounding box regression and box classification.

In Fast R-CNN, the proposal windows were fed into a classifier neural network. However, the approach suffers from a fundamental drawback that arises from the large number of candidates that need to be processed by a separate neural network, resulting in increased computational complexity. Due to this drawback, the author of Faster CNN focused on suppressing it.

Before explaining the principle of classification in Faster R-CNN, we describe the approaches used for multiscale prediction. In [159] three different options are presented. The first is based on a pyramid of images, where the input image is resized to different scales and for each size a feature map is computed, which is time-consuming. The second method preserves the size of the image/feature map and uses sliding windows of different sizes to extract features. Finally, the third option, used by Faster R-CNN, is called pyramid of anchors and relies only on feature maps and images of a single scale/size. This approach replaces the direct bounding box regression with a method based on the localization of the bounding box relative to the anchors. It is appropriate to mention that this method is used in other state-of-the-art algorithms that outperform the Faster R-CNN [17][160].

From the perspective of Faster R-CNN architecture, the first part employs a convolutional neural network that is intended for generating object proposals and performing objection detection simultaneously. This stage of the neural network utilizes the same weights to solve both of the tasks.

[7] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
[8] http://host.robots.ox.ac.uk/pascal/VOC/voc2010/
[9] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

The output of the feature extractor is then sequentially slid over by a small neural network. In the published research paper, the small network contains convolutional layers followed by two sibling 1×1 convolutional layers.

Using this small neural network, the prediction is conducted at each location in the image. Thus, the vector size for regression is derived from the number of detections k, assuming that the number of detections per location is equal to the number of anchors. The bounding box is defined by coordinates and size relative to an anchor (c_x, c_y, c_w, c_h). It follows that the vector size is $4k$ and, for classification, the vector size is $2k$ due to the use of a two-class softmax layer.

Unlike Fast R-CNN, it is clear that Faster R-CNN is a translation-invariant detector, as there is the same set of anchors for each sliding window. When using either R-CNN or Fast R-CNN algorithm, we cannot guarantee that the same proposal will be generated at different locations. Due to the irregular distribution of regions produced by the MultiBox/SelectiveSearch generator [158] in an image, it is challenging to ensure translation invariance of the detector.

At training time, a label with IoU between the predicted box and the ground truth box above 0.7 is considered a positive label. On the other hand, the labels with the IoU threshold below 0.3 are considered negative.

The loss function is divided into two parts, where the first is the classification logarithmic loss and the second is the regression loss based on a robust logarithmic function (smooth L_1, see Eq. 3.14). The loss is defined by the following equation [159]

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right). \tag{3.15}$$

where p_i is the predicted probability and t_i is the predicted vector described by four parameterized coordinates for each anchor indexed by i. The parameters p_i^*, t_i^* express the ground truth labels for the classes and for the box associated with a positive anchor, respectively. The L_{cls} performs log loss over two classes. For regression loss, it is obvious that the function is activated only for positive anchors and the loss is defined as $L_{reg}\left(t_i, t_i^*\right) = R\left(t_i - t_i^*\right)$, where R is the robust loss function (smooth L_1). The normalization of the losses is expressed by the parameters N_{cls} and N_{reg}, while the loss is balanced by λ.

3.4.4 Single Shot Detector

The *Single Shot Detector* [160] (SSD) belongs to a group of one-stage detectors designed for real-time object detection and classification. Although the use of improved versions of the R-CNN has led to a reduction in processing time, they are still not suitable for use in real-time applications. Therefore, the goal of the next generation of detectors is to suppress this drawback. The authors of [160] modified the extraction method to reduce the number of region evaluations.

In the SSD, region localization is provided by a multi-scale feature map and default boxes. Although the SSD is considered a single stage detector, the architecture of the neural network is divided into two subnetworks, the backbone and the head. In many neural networks, a pre-trained backbone is used to extract the features of common objects. However, the backbone usually has a fully connected layer (FC) as the output layer. To adapt it to the detector, the FC layer is removed and replaced by the head. Among the well-known backbones, ResNet, DenseNet, AlexNet and VGG16 can be used for these purposes (for more information, see Section 3.3). The solution published in the original paper uses VGG16.

From an architectural standpoint, the backbone output is used in different stages of the SSD head network. For a better understanding, the architecture is shown in Figure 3.15. Let us look at the first stage of the SSD head; the output of $Conv4\_3$ preserves the same shape (38×38) as the output of the backbone. Each stage of the head is divided into cells, and these cells are responsible for classification and localization.

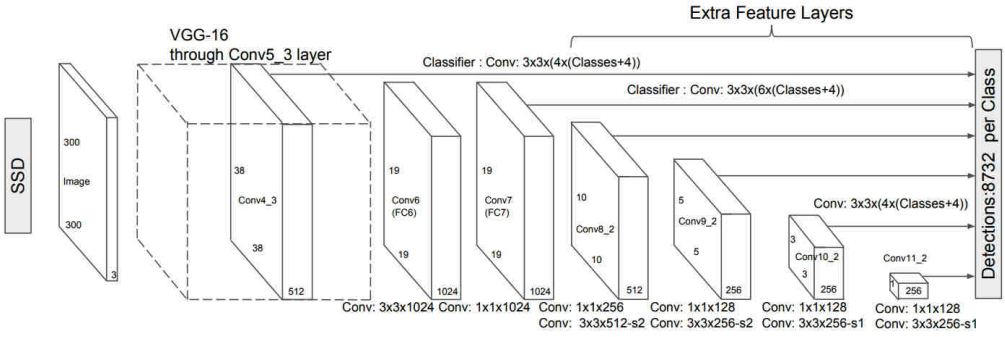


Figure 3.15: Architecture of Single Shot Detector [160].

As mentioned above, the SSD detector uses predefined bounding boxes called default boxes. Each cell is assigned four default boxes. Now, one might ask why it is necessary to use a default box when we have bounding boxes derived from annotations. The idea is to suppress the influence of the shape of the ground truth object on the training process, since it can be influenced by instances of an object with different shape. The predefined set of default bounding boxes is the same for each cell in a given layer. In addition, the set of default boxes can be different for each of the head layers.

Overall, the predicted bounding boxes are calculated relative to the default bounding boxes at each cell. The bounding boxes are represented by four values (c_x, c_y, w, h), where c_x and c_y are the differences between the center of the bounding box and the center of the default bounding box. Given this knowledge, it is clear that a default bounding box represents only one potential detection. The default boxes are chosen manually to find the sizes and proportions that most closely resemble the real objects.

Unlike most two-stage detectors, SSD employs only a single CNN to generate proposals and perform their classification. Additionally, the output of the backbone is segmented into cells, with individual cells responsible for localization and classification. Moreover, the effectiveness of multiscale detection is tied to the number of head stages. This is the only reason why SSD uses multiple layers with different resolutions. The layer closest to the feature extractor is responsible for detecting small objects, because it has a higher resolution. While the following layers have gradually decreasing resolutions and are responsible for detecting larger objects. In the paper [160], the detector head consists of six convolution layers, where five of which perform object detection. In each layer, convolution is conducted using small filters and its output is fed to the next stage and can also contribute to the output vector. The shape of the contributions is defined by the number of classes, the number of default boxes, and the size of the bounding box vector. By changing the part, we can adapt the SSD according to the specific use case.

In practice, the probability that the ROI labels of training samples have the same shape as the default bounding box is close to zero. This raises a question of how to determine which detections should be labeled as positive. In SSD, the IoU metric was used to de-

termine the ratio of the intersected area to the connected area for two regions. Based on the published parameters, if IoU is greater than 0.5, the match is considered positive.

In order to train a neural network, it is necessary to define a loss function. As we already know, the SSD performs localization and classification. Therefore, the loss function consists of two parts, the localization loss L_{loc} and the confidence loss L_{conf} [160], defined as follows [160]

$$L(x, c, l, g) = \frac{1}{N} \left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \right) \tag{3.16}$$

where N is the number of the matched default boxes. The loss function L_{loc} is defined by [160]

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{c_x, c_y, w, h\}} x_{ij}^{k} \operatorname{smooth}_{\mathrm{L1}} \left(l_i^m - \hat{g}_j^m \right) \tag{3.17}$$

where $x_{ij}^{p} = \{1, 0\}$ determines the correspondence between the i^{th} default box and the $j^t h$ ground truth box of category p. The predicted box is denoted as l and the ground parameter is denoted as g. Additionally, the confidence loss is a cross-entropy loss over categories c given by [160]

$$L_{\mathrm{conf}}(x, c) = -\sum_{i \in \mathrm{Pos}}^{N} x_{ij}^{p} \log \left(\hat{c}_i^p \right) - \sum_{i \in Neg} \log \left(\hat{c}_i^0 \right), \quad \text{where} \quad \hat{c}_i^p = \frac{e^{c_i^p}}{\sum_p e^{c_i^p}} \tag{3.18}$$

In order to ensure the stability of the neural network training, the authors use support methods. The first one is focused on the balance ratio between the negative and the positive matches and it is called the *hard negative mining* [160]. Instead of using all negative examples, only a subset of them is selected for training. In practice, the negative examples are sorted in ascending order by their confidence loss for each default box. Such sets are then selected from the top to achieve a 3:1 ratio of negative to positive examples. An equally important part of SSD training is data augmentation, which helps to improve the accuracy of the neural network.

3.4.5 Multi-task Cascaded Convolutional Networks

In contrast to a single-task neural network, a multitask neural network brings the ability to solve more than one task, which can be useful in many cases. Similarly, in face recognition, we can fuse face detection and face keypoint detection into one step using the multitask neural network. One of the representatives is *Multi-task Cascaded Convolutional Networks* (MTCNN) [161], which has the ability to perform detection and landmarks regression simultaneously. The process is carried out in several stages. Before the input is processed by neural networks, it is resized to multilevel scale pyramid to ensure the scale invariance of the detector.

The pyramid is then fed into the first stage, called *Proposal Network* (P-Net) [161], which is built by a fully convolutional neural network, so its output is a feature map. Similar to the feature extractor in Faster R-CNN, the ROI candidates are generated by the P-Net. Then, the NMS algorithm reduces the obtained candidates calibrated by the bounding box vector.

The next stage, the neural network called *Refine Network* (R-Net) [161], is intended to suppress a large number of false proposals, and then, like in the previous stage, the candidates are reduced by the NMS. Unlike the P-Net, the R-Net is fully connected, since its outputs are represented by a vector.

The third stage, known as the O-net network, excludes other candidates from the remaining ROIs and is responsible for determining the output of the detector. In the original paper, the detector has three outputs: face classification, bounding box regression, and face landmark regression.

At training time, each task is assigned a specific loss function. For face classification, which is considered to be a binary classification, the cross-entropy loss function is used, given by [161]

$$L_i^{det} = -\left(y_i^{det} \log\left(p_i\right) + \left(1 - y_i^{det}\right)\left(1 - \log\left(p_i\right)\right)\right) \tag{3.19}$$

where the p_i expresses the probability of the presence of a face.

Another loss function is intended for bounding box regression, specifically to compute the offset between the predicted bounding box and the nearest ground truth [161]

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2 \tag{3.20}$$

where the $\hat{y}_i^{box}$ is the output of the network and y_i^{box} is the ground truth coordinate.

Finally, the last loss function, similar to the bounding box regression, calculates the difference between ground truth landmarks and predicted landmarks. Mathematically, it is defined as follows

$$L_i^{\text{landmark}} = \left\| \hat{y}_i^{\text{landmark}} - y_i^{\text{landmark}} \right\|_2^2 \tag{3.21}$$

where $\hat{y}_i^{landmark}$ expresses the regression of the landmarks and y_i^{box} is the location of the ground truth of the landmarks. Both of these functions use the Euclidean distance as a metric.

Overall, the loss function is defined by [161]

$$L = \min \sum_{i=1}^{N} \sum_{j \in \{\text{det,box,landmark}\}} \alpha_j \beta_i^j L_i^j \tag{3.22}$$

where N is the number of samples, α_j expresses the importance of the task and $\beta_i^j \in \{0, 1\}$ denotes negative or positive samples, respectively.

Another technique closely related to neural network training is the hard sample mining. The technique is used to mine only those samples that can significantly contribute to the training within a mini-batch. In the case of MTCNN, the samples are sorted by their losses calculated in the forward propagation phases. Of these, 70 % is used to compute the gradient.

According to the published paper [162], the CNN for face detection performs binary classification, which means that we need fewer parameters in the neural network while maintaining higher discrimination power. The authors have compared their architecture with that of [162] and found that the response produced by their solution provides better results.

3.4.6 YOLO

YOLO (You Only Look Once) [163] stands as a renowned neural network for object detection and serves as another representative of one-stage detectors. Similar to the SSD, the classification and box regression tasks are performed simultaneously by the YOLO architecture. The output of the feature extractor is divided into cells, where each cell is responsible for detecting an object. Unlike R-CNN and Fast R-CNN, YOLO views the image globally so that contextual features can be utilized. YOLO outperforms two-stage detectors such as R-CNN, Fast R-CNN, and Faster R-CNN. Neverthless, the comparison between SSD and YOLO is not as straightforward, because it must take into account the version of YOLO and the application for which the detector is used.

On closer inspection, the architecture of YOLO includes two main parts. The ImageNet [163] was chosen as the default backbone, although other architectures can be used for feature extraction.

With the head of the YOLO neural network, the feature map is divided into a S×S grid, where S is a size of one dimension. In the published article, the grid has a size of 7×7 cells. For each cell, vectors are defined consisting of B bounding boxes with their confidence scores and C class probabilities. A bounding box is represented by four predictions x,y,w,h and its confidence score, which determine how probable it is that an object was detected and how accurately it is defined and located. A bounding box is assigned to a cell if its center is inside the cell, otherwise the confidence score is set to zero. During the testing phase, the confidences of the bounding boxes are multiplied by the probabilities of the class C (in each cell there is only one set of probabilities for the class), and the result predicts the assignment of the object to a particular class. Using the coefficients $S = 7$ and $B = 2$, the neural network can detect up to 98 boxes.

In the YOLO loss function, a squared metric is used for both regression and classification of a bounding box to determine the error between the reference and the prediction. Furthermore, the part of the loss that solves the regression of a bounding box is weighed by λ_{coord}, while λ_{noobj} is related to confidence predictions for boxes that do not contain objects. Balancing these parameters can lead to model stability. According to the publication, the authors increase the weight for the prediction of a bounding box at the expense of λ_{noobj}. Overall, the loss function L is defined by the following equation [163]

$$
\begin{aligned}
L = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
\tag{3.23}
$$

where by $\mathbb{1}_i^{\text{obj}}$ determines whether the object appears in cell i or not. Similarly, $\mathbb{1}_i^{\text{obj}}$ determines whether the box predictor j in cell i is "responsible" for the prediction.

We have described the techniques that were used in the first version of YOLO. Despite its numerous advantages, this network also has its shortcomings, such as the limitation of detection per cell, since it can only perform two detections per cell for one class. This means that small objects in group cannot be detected properly. Another limitation is related to the loss function and the fact that the calculation of error is the same for different object sizes. Therefore, the larger object is not susceptible to small errors, while small objects should be significantly affected by small errors (they may have a low IoU value).

Since the YOLO detector is a very popular detector, its subsequent versions have improved performance and suppressed drawbacks. The following sections briefly describe the selected newer YOLOs.

YOLO9000

Based on the YOLOv1 error analysis, the authors redesign the YOLO architecture by adding a batch normalization layer to the convolution layers [164]. This increases the mAP value by more than 2 % and also helps to regularize the model. Unlike YOLOv1, this YOLO predicts detections on a 13×13 feature map, whereas the preceding pass-through (26×26) layer contributes to the output vectors consisting of predicted detections. This modification is intended to improve the ability of the neural network to detect smaller objects. Another improvement was achieved by using anchor boxes instead of directly predicting the bounding boxes. Overall, we can see that the number of units is the same, but the topology of the layer was changed, i.e. the spatial locations are stacked in different channels.

YOLOv5

Since the release of the original version of YOLO, several other versions have been released through 2020 [165]. In 2020, just a few months after the release of YOLOv4 [166], Ultralytics introduced YOLOv5. YOLOv5 replaces PyTorch's darknet implementation and is heavily inspired by the techniques introduced in YOLOv4. A notable architectural change is the use of a backbone, neck, and head-based structure first introduced in YOLOv4. This architectural principle is also described in RetinaFace (Section 3.4.7).

The underlying architecture is rooted in *CSPDarknet53* [166], which has been adapted to lower memory and computational costs. For the neck, the Spatial Pyramid Pooling Fast architecture is used together with a modified CSP-PAN. The design of the heads is inspired by YOLOv3 [167].

In addition, YOLOv5 incorporates several augmentation methods, including Mosaic, Copy Pass, Random Affine, MixUp, HSV augmentation, and others, which are taken from the albumentations package[10].

Similar to YOLOv4, the training uses the *CIoU loss* [166] for supervision and uses *cross mini-batch training*.

YOLOv7

YOLOv7 [168] introduced further changes to the architecture and subroutine algorithms to provide better training. The underlying architecture is also based on YOLOv4 [166]. The primary modification is the integration of *Extended efficient layer aggregation network*

[10] https://albumentations.ai/

(E-ELAN) [168], a mechanism that manages the shortest and longest gradient paths, allowing for more efficient learning and convergence of the network. In addition, the architecture is designed with scalability in mind, which is achieved through a concatenation-based structure.

The *trainable bag of freebies*[11] category includes techniques such as RepConv [169], where the identity connection is removed, resulting in RepConvN, specifically adapted for YOLOv7. For training support, two types of heads are used: one for final output and another designed to enhance the training process. These heads are associated with two different sets of labels. Soft labels are associated with the auxiliary head, while fine labels correspond to the output head. In addition, YOLOv7 has placed the batch normalization in the Conv-Bn activation to connect a convolutional layer directly to the batch normalization layer. Similar to YOLOR [170], this version of YOLO uses implicit knowledge to avoid suboptimal future generation. The final reference model, YOLOv7, is marked as an EMA model intended for the teacher method.

3.4.7 RetinaNet and RetinaFace

RetinaNet [171] is one of the state-of-the-art neural networks designed to solve multiple tasks in a single step. From a face detection standpoint, this architecture is used in the derived network called RetinaFace [154]. RetinaFace can solve tasks involving face classification, face box regression, face landmark regression, and dense face regression. However, the main concept is the same for both RetinaNet and RetinaFace, differing only in their configurations.

Let us examine the architecture of RetinaNet. In the first part, the input image is processed by ResNet, and selected branches of its outputs are fed into a multiscale feature pyramid based on the *Feature Pyramid Network* (FPN) [172]. To understand the use of the FPN, it is necessary to understand the meaning of the layers in context. As the size of the feature layer decreases, the ability to extract a semantic value increases. In the SSD, only the lower layers are used for object detection, so the detector performs much worse for small objects. The idea of FPN is to add a complement to ResNet that combines low-resolution (semantically strong) features with high-resolution features. In other words, the FPN performs an upscaling of the context information represented by the feature layer and merges it with the corresponding feature map of the backbone. Overall, the FPN is fast in computation and semantically strong.

In this architecture, instances of a subnetwork responsible for performing the multi-tasks is assigned to selected levels of the pyramid. In RetinaNet, this part is called the *context module* and is responsible for solving the tasks mentioned above.

The loss function responsible for the classification is designed with the focal loss instead of the cross-entropy loss used in most detectors. The cross-entropy loss is not suitable in a situation where the classes are highly unbalanced, which occurs when the number of anchors is high. The focal loss allows to increase the importance of hard samples to the loss function and vice versa to reduce the influence of easy samples. The focal loss is described as [171]

$$\mathrm{FL}\left(p_{\mathrm{t}}\right) = -\left(1-p_{\mathrm{t}}\right)^{\gamma} \log\left(p_{\mathrm{t}}\right) \tag{3.24}$$

[11] The term *bag of freebies* characterizes methods that modify training strategies or increase training costs, all in an effort to improve object detector accuracy while keeping inference costs the same [166]

where p_t is the estimated probability depending on the ground truth class and γ is the focusing parameter. However, the loss function was not used in the published RetinaFace. The definition of the RetinaFace loss function is as follows [154]

$$L = L_{cls}\left(p_i, p_i^*\right) + \lambda_1 p_i^* L_{box}\left(t_i, t_i^*\right) + \lambda_2 p_i^* L_{pts}\left(l_i, l_i^*\right) + \lambda_3 p_i^* L_{\text{pixel}} \tag{3.25}$$

where L_{cls} is the class probability for the anchor i realized by the softmax loss. L_{box} refers to loss for bounding box regression, where the bounding box is represented by $t_i = \{t_x, t_y, t_w, t_h\}_i$ and its ground truth is marked as t_i^*. Furthermore, we use $L_{box} = R\left(t_i - t_i^*\right)$, where R is the robust loss function $(Smooth - L_1)$. The loss of facial landmarks $L_{pts}\left(l_i, l_i^*\right)$ is solved as a regression task, where $\{l_{x_1}, l_{y_1}, \ldots, l_{x_5}, l_{y_5}\}_i$ and $l_i^* = \left\{l_{x_1}^*, l_{y_1}^*, \ldots, l_{x_5}^*, l_{y_5}^*\right\}_i$ defined the prediction of five face landmarks and their ground truth for positive anchor. The last part of the loss is a dense regression that includes the predicted parameters P_{cam}, the illumination parameters P_{ill}, and the colored mesh $\mathcal{D}_{P_{ST}}$. Then it is defined by [154]

$$L_{\text{pixel}} = \frac{1}{W * H} \sum_{i}^{W} \sum_{i}^{H} \left\| \mathcal{R}\left(\mathcal{D}_{P_{ST}}, P_{\text{cam}}, P_{ill}\right)_{i,j} - I_{i,j}^* \right\|_1 \tag{3.26}$$

where W and H are the dimensions of the anchor crop $I_{i,j}^*$ (ground truth).

3.4.8 TinaFace and SCRFD

An alternative approach to designing neural networks for face detection involves tailoring the model exclusively for the face detection purpose. An example of such network is TinaFace [155]. TinaFace is a detector based on established state-of-the-art techniques. Its creators used the principles of the RetinaNet neural network, widely regarded as the most advanced model for generic object detection available today.

TinaFace and RetinaFace share a common core architecture, but instead of using *context blocks* [154], the TinaFace's authors use *inception blocks* [155] combined with head blocks. Inception blocks consist of convolutional layers in a parallel topology used for both object and context capture at multiple scales [155]. In addition to classification and regression heads, TinaFace also incorporates an IoU-aware head, which is used for resorting the classification score and suppressing false-positive detected boxes. TinaFace's feature extractor utilizes a six-level FPN, which operates on square-shape images, each with dimensions of (i.e 640×640).

The TinaFace's authors also introduced Distance-IoU Loss (DIoU), which outperforms Smooth L1 loss for bounding box regression. While Smooth L1 loss guarantees stable training, this metric does not consider IoU. Therefore, the authors replace Smooth L1 with DIoU, which, according to them, is more suitable for small objects. DIoU is defined as follows [154]

$$L_{DIoU} = 1 - IoU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} \tag{3.27}$$

where b and b^{gt} express the central points of the predicted box and the ground truth box, ρ^2 is the L2 distance, the C means the diagonal length of the smallest enclosing box covering the two boxes.

Although the performance of TinaFace is really impressive, the authors made the architecture based on the analysis of WIDER Face dataset, in which roughly two thirds of the data represent small objects.

The next iteration of RetinaNet-like algorithms is SCFRD, *Sample and Computation Redistribution for Efficient Face Detection* [173]. The authors perform an analysis of TinaFace and propose improvements in the design of face detection, both from an architectural and a training standpoint.

Though analyzing neural network model space, the authors find a model with reduced complexity, whereas the performance was preserved. This optimization is performed using a technique that is used in RegNet [174]. The idea is based on constricting selected degrees of freedom of the model. Simply, some parameters of neural network are fixed, and some can be changed. For example, we fix filter kernel size and define that the number of filters is between 16 and 64 where the number must be divisible by sixteen. From this we can generate four models. Using the techniques, we can deal with the influence of the parameters to performance of training network.

The training process was also adjusted considering that the WIDER dataset contains a large number of small faces.

3.4.9 Summary

Based on the study of face detectors, we can consider detectors with the FPN architecture, which is a fundamental part of RetinaFace, TinaFace and SCFRD, as state-of-the-art detectors. These detectors together with the YOLOv7 detector represent the best solution for face detection today. In future developments, we expect reduction of the number of parameters and acceleration of the response time of these neural networks.

3.5 Face recognition algorithms based on deep convolutional neural network

According to the timeline presented in Section 2.1.1, modern algorithms for face recognition have been based on neural networks since around 2015. In addition, sufficient computing power and large datasets with face images have become available. These aspects have led researchers to focus on solutions based on neural networks. The first published neural network was DeepFace, which achieved a recognition accuracy of 97.35 % in face verification on the LFW dataset [122], while the accuracy obtained by humans was 97.53 %. Although the results of face verification were slightly worse in the case of the neural network, this started the era of face recognition using neural networks [175]. Currently, face recognition algorithms based on neural networks achieve excellent accuracy.

To train a neural network, it is necessary to use a dataset meeting the criteria defined in Section 3.2. However, in practice, it is difficult to meet the criteria, mainly because the images in the dataset may be mislabeled or corrupted. As the amount of data increases, it becomes increasingly difficult to maintain an error-free dataset. However, the dataset issue is not the only problem we have to deal with. Another problem is how to properly train a neural network to correctly determine the identity of a person. According to the theory of neural networks, face recognition is the task of classifying a face image as belonging to a certain identity. In the classification tasks, the classes are represented as the output of the last layer.

However, there is a fundamental difference compared to common classification tasks. What happens if we need to recognize a person on whom the neural network has not been trained? The output layer of the neural network can be extended by a new class, and then the neural network has to be retrained. This approach is very impractical and difficult to

apply in practice. Therefore, it is necessary to use a method in which the neural network does not need to be retrained. In general, this can be solved by a neural network that can map an input face image to a unique vector even for unknown identities.

In the following section, we describe known face recognition algorithms based on a neural network. The loss functions mentioned are based on the loss functions presented in Section 3.1.

3.5.1 Face embedding

To compare pairs of face images, it is necessary to transform the face images into another representation in such a way that even if the input images for the same person are taken in different positions and various environments, they should still be mapped to the same place in space. One way to achieve this is to use an operation called *embedding*. In mathematics, an embedding is a representation of a topological object in a given space in such a way that its connectivity or algebraic properties are preserved [175]. The embedding of an instance X is given by a injective map x, $(x \to X)$.

From the neural networks standpoint, the embedding produced by a deep neural network mainly transforms an input into a vector x in the Euclidean space $x \in \mathbb{R}^n$, where n denotes the dimensionality of the space. The part of the neural network responsible for embedding is called feature extractor (Section 3.3 and solves the above problem. In general, it can be said that face embedding is used by the vast majority of face recognition algorithms. However, this approach has other advantages, including the ability to quickly compare the vectors. The similarity between the vectors can usually be determined by the *Euclidean distance*, the *cosine similarity* [176] and the *cosine distance* [177]. These metrics are composed of several mathematical operations, making them efficient to compute. The Euclidean distance, also known as the L2 distance, is the length between two points in Euclidean space and can be computed by [178]

$$d(A, B) = \sqrt{\left(\sum_{i=1}^{N} (A_i - B_i)^2\right)} \tag{3.28}$$

where A and B are the vectors $(A, B \in \mathbb{R}^n)$ and d is the Euclidean distance.

The cosine similarity is the angle between two vectors and can be computed as follows [179]

$$cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.29}$$

where A and B are vectors and θ is cosine similarity.

The cosine distance is directly related to the cosine similarity and is defined as [177]

$$d(A, B) = 1 - \frac{A \cdot B}{\|A\| \, \|B\|} = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.30}$$

You may be wondering how to choose appropriate metrics? The choice depends mainly on the loss function used. If the vectors have an angular distribution, it is better to use cosine similarity or cosine distance.

The challenge in face embedding is to create a unique vector for each identity with minimal intra-class variability and maximum distance from vectors representing other classes.

Therefore, the neural network must be trained to minimize the distance between samples of the same class and maximize the distance between samples of the different classes. In recent years, algorithms with various loss functions have been published. Recently, it can be said that the design of the loss function [180], the acceleration of training and the enlargement and filtering of the dataset are the main directions of the development of face recognition algorithms. Following [181], we describe several neural networks for face embedding.

3.5.2 DeepFace

DeepFace,introduced in 2014 [87], can be regarded as the first dedicated neural network for face recognition embedding. Although the crucial part of the algorithm is the neural network, other machine learning algorithms are utilized for preprocessing input of the neural network and for classifying the output of the neural network. As preprocessor is considered the part that is responsible for face alignment. The need to align the face in an image arises from the nature of the data obtained under unconstrained scenarios.

In DeepFace, face alignment is performed using both 2D face alignment and 3D mapping techniques. 2D alignment provides a basic in-plane alignment that changes image rotation, scale, and translation through the transformation matrix. However, 2D alignment cannot compensate for out-of-plane face rotation. Also, a face shape cannot be considered as a generic 3D object, such as a sphere. Therefore, the authors implemented a suitable method to perform 3D alignment while preserving identity factors. In DeepFace, the 3D alignment is based on the generic 3D head mapping to the 2D plane by the affine camera P.

Thus, in the next step, the 67 face landmarks are assigned to the corresponding landmarks in the generic 3D model. Furthermore, the points are transformed into a 2D image using the camera matrix P composed of the estimated vectors. Since the camera matrix P is only an approximation that does not take into account all elements of the perspective projection, it is necessary to modify each reference landmark point x_{3d} by its corresponding residuals r, which are determined during the fitting of P. The modified points are denoted by $\tilde{x_{3d}}$. Finally, the frontalization is performed by a piecewise affine transformation T from x_{2d} (source) to $\tilde{x_{3d}}$ (target).

The workhorse of the DeepFace is the feature extractor, which is based on a deep neural network. As mentioned in Section 3.5.1, we consider face recognition problem as a multiclass classification task. Therefore, the goal of training the neural network is to maximize the probability of assigning a correct class to an identity, which is achieved using the cross-entropy loss function and the SGD optimizer (Section 3.1).

In perspective of architecture, the model of neural network consists of the convolutional layers, max pooling layers and fully connected layers. In the last fully connected layer, the individual neurons are assigned to the corresponding identities, while the penultimate layer is composed of neurons that have ability to describe the face characteristic in a vector form. Of course, the vector is well known as a face embedding vector (Section 3.5.1).

In mathematical form, the output representation $G(I)$ is derived from the preprocessing transformation and the output of a feedforward neural network. Thus, the representation can be described as $G(I) = g_\phi^{F_7}(g_\phi^{L_6}(...g_\phi^{C_1}(T(I, \theta_T))...))$ where $\phi = \{C_1, ..., F_7\}$ are parameters of the network (Figure 3.16) and $\theta_T = \{x_{2d}, \vec{P}, \vec{r}\}$ are the face alignment parameters.

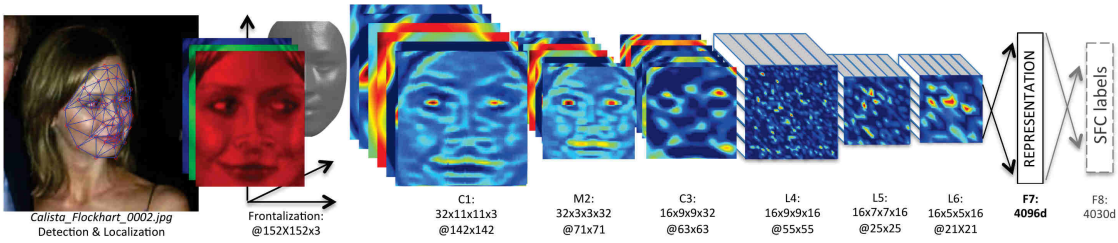


Figure 3.16: Architecture of DeepFace neural network [87].

To compare feature vectors, the DeepFace algorithm uses the weighted χ^2 distance that is defined as

$$\chi^2(f_1, f_2) = \sum_i \frac{w_i(f_1[i] - f_2[i])^2}{(f_1[i] + f_2[i])} \tag{3.31}$$

where w is the weight parameter that is obtained by the SVM classifier and that is applied to the vector of elements $(f_1[i] - f_2[i])^2/(f_1[i] + f_2[i])$ and $f_{1,2}$ are the DeepFace representations.

The paper then describes the modification of the neural network called the *Siamese neural network* [87]. This neural network determines whether the two images belong to the same person or not. Although the neural network has shared weights, the computation has to be done twice (for both input images). The distance between two feature vectors is defined as [87]

$$d(f_1, f_2) = \sum_i \alpha_i \|f_1[i] - f_2[i]\| \tag{3.32}$$

where α_i is the trainable parameter.

To achieve better accuracy, the final published results are based on the composition of the outputs of three neural networks. Except for the Siamese network, each of the networks has the same architecture, but the their inputs differ due to the fact that they are trained with different seeds. The predicted distance is weighted by the weight vector determined by the non-linear SVM. Overall, the final method achieves a face recognition accuracy of 97.35 % on the LFW dataset, compared to the accuracy of 97.53 % obtained by a human. With DeepFace, the new generation of face recognition algorithms has begun.

3.5.3 FaceNet

The second popular method of face recognition based on neural networks was introduced by Google researchers in 2015 and is called *FaceNet* [17]. Using face embedding, the neural network maps a face image x to the feature space R^n. In FaceNet, the output vector v is composed of 128 real values, which means that the identity is represented in a 128-dimensional hyperspace. Unlike the DeepFace neural network, FaceNet training is supervised by an innovative loss function called triplet loss. While the cross-entropy utilizes class probabilities, triplet loss uses the distance metric, specifically the Euclidean distance metric, see Section 3.17.

Let us take a closer look at the triplet loss function. In general, the loss function is used in machine learning to determine the error between the values produced by an algorithm and the required target values. The idea of a triplet loss function is based on minimizing

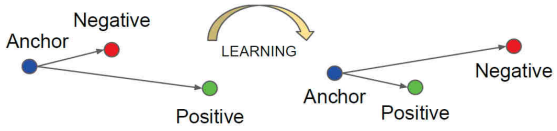


Figure 3.17: Principle of the triplet loss function [17].

the L2 distance between the faces of the same identity and enforcing maximal distance between the faces of different identities. The algorithm is similar to the nearest neighbor classification. During the computation, it is necessary to establish the anchor for each triplet to which positive and negative samples are related. Then the triplet loss ensures that the positive sample x_i^p (positive) is close to a face image x_i^a (anchor) compared to a face image x_i^n that should be far from the face image x_i^a. In mathematical terms, with the loss function, we want the loss function to satisfy the following condition [17]

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2, \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau \tag{3.33}$$

where τ is the set of all possible triplets, α expresses the coefficient to ensure the margin between the negative and positive pairs. Similarly, the triplet loss function L is defined as [17]

$$L = \sum_N^i \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{3.34}$$

where the N expresses the cardinality of the training dataset.

However, it is not feasible to generate all possible triplets. Therefore, it is crucial to select relevant sets of triplets that contribute to the improvement of the model. Such triplets are called hard triplets. Consequently, it is difficult to compute *argmax* and *argmin* for all samples in a data set, calculated by [17]

$$argmax_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2, argmin_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2. \tag{3.35}$$

for both x_i^p (*hard positive*) and x_i^n (*hard negative*) related to anchor x_i^a

There are two ways to avoid this problem: generating the triplets online and generating the triplets offline. When generating the triplets offline, the last checkpoint is used every n steps to calculate *argmin* and *argmax* on a subset of the data. Another approach is to choose the subset by selecting hard positive/negative examples from a mini-batch. Commonly used option is to generate the triplets online. However, the selection of triplets within the mini-batch is still an essential procedure. The proposed method uses a mini-batch consisting of 20 face images per identity, complemented by random negative examples. In the set, all anchor-positive pairs are used, while the set of used negative examples is reduced. Instead of hard samples, which can lead to bad local minima, the set of semi-hard samples is used, which satisfies the following condition [17]

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2. \tag{3.36}$$

In the paper, the authors proposed two neural network architectures. The first architecture consists of basic elements for convolutional neural networks, such as a convolutional

layer, a pooling layer and a fully connected layer. However, the second (NN2) includes a different type of blocks, the inception block [145], for more information see Section 3.3.

While the first architecture, denoted NN1, has 140 milion parameters, the second, based on GoogleNet inception models, has 20 times fewer parameters than the first one. For the triplet loss function, the gradient descent of the neural network was performed using the SGD optimizer. It is interesting to note that the authors state that the training time was between 1,000 and 2,000 hours.

Unlike DeepFace [87], the presented approach does not require any post-processing algorithms. Therefore, the only task to improve performance is to optimize the loss function. Other strengths of the presented approaches are less strict conditions for face alignment. In particular, on the LFW dataset, the classification accuracy of the algorithms is 98.87 %±0.15 when the face is centered in the images and 99.63 %±0.09 when the face images are aligned.

3.5.4 VGGNet

As mentioned in Section 3.3, backbones are essential components of face embedding neural networks, responsible for extracting characteristic features from images. In their paper [112], the authors introduced a methodology for constructing a dataset by collecting face images from public sources. Additionally, they designed and trained various neural networks using different backbones, all based on VGG nets (where VGG stands for *Visual Geometry Group*).

There are five steps to the data collection approach presented in their paper. The goal of the first step is to obtain a list of candidate names, where each name represents an identity that will be used in the following steps. This is accomplished through the processing of the Internet Movie Data Base (IMDB)[12]. The next step is to intersect the list of candidate names with all the people in the Freebase knowledge graph [127]. The results included 5k identities for their 200 images downloaded via Google Image Search[13].

The images are initially filtered by human annotators to assess their purity, reducing the number of identities to 3,250. Subsequently, these identities are compared with those present in the LFW and YTF datasets. If they match, they are excluded. The remaining identities are supplemented by searching for additional corresponding images.

The aim of the third stage is to automatically remove any erroneous faces to improve purity. After this, the datasets contain 1,000 images per identity. Since the same images from different sources can be present in the dataset, the fourth step deals with the removal of duplicates. In the last step, the images are again manually filtered. Seeing that manual annotation is time-consuming, the authors trained AlexNet [141] to classify the images according to their identity. In their paper, the proposed dataset was again validated by human annotators in the final step.

From an architectural point of view, the authors consider three different architectures with an input size of 224×224. In addition, the input is normalized to ensure the stability of the optimization algorithm.

The procedure for obtaining a neural network for face embedding has already been established in the preceding sections. This includes removing the last fully connected layer. To control the training, the author used the triplet loss function introduced in FaceNet, see Section 3.5.3.

[12] https://www.imdb.com
[13] https://www.google.com

In conclusion, the authors selected and evaluated the best architecture so that its accuracy on the LFW is 98.98 %.

3.5.5 From Softmax loss to Center Loss

From a machine learning perspective, the goal of face recognition is to assign a label or identity to an input sample. Therefore, face recognition can essentially be seen as a multiclass classification task. When developing a face recognition application, we can rely on classifier algorithms or neural networks, which are designed to solve similar types of machine learning tasks. While the performance of neural networks depends on various factors, researchers primarily focus on optimizing the loss function to supervise the training of a neural network.

Let us go back to the perspective of looking at face recognition as a cross-entropy activation function task. The basis for this task is the softmax activation function. In the next sections, we focus on this activation function and analyze it.

Softmax loss

From the given information, it is obvious that each class is represented in the output vector. Unlike the sigmoid activation function, the softmax loss produces a distribution in which the sum of the values in the vector equals one [182]. For face recognition, the softmax loss function is defined by [183]

$$L_{softmax} = -\sum_{i=1}^{m} log \frac{exp^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} exp^{W_j^T x_i + b_j}} \tag{3.37}$$

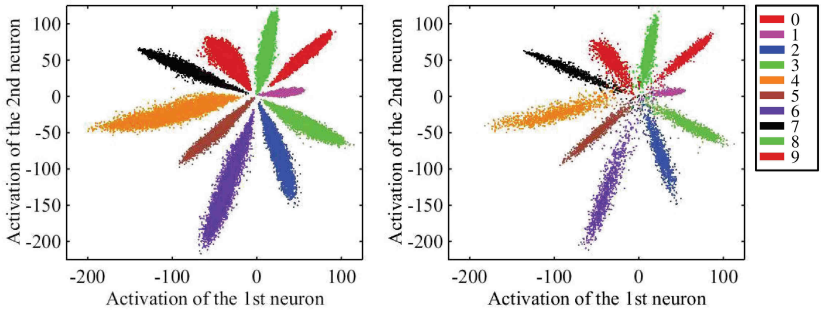
where $W_{\{j,y_i\}}^T x_i + b_{\{j,y_i\}}$ refers to the last fully connected layer. The $x_i \in \mathbb{R}^d$ expresses the y_i^{th} deep feature belonging to the y_i^{th} class. Next, $W_j \in \mathbb{R}$ is the weight matrix column $W \in \mathbb{R}^{d \times n}$, $b \in \mathbb{R}^n$ is the bias term, m is the number of training samples and n is the number of classes.

A possible area for research is the observation of the distribution of feature vectors in a space. Therefore, in [183], an experiment focused on evaluating the distribution of vectors provided by a neural network trained using softmax loss was conducted. For the experiment, a model based on the LeNets neural network was designed [184]. The output of the model consists of only two output neurons, in which their responses are then used to observe the distribution of the features.

The visualization of the responses shows that the learned features are separable, while they are less discriminative due to low intra-class variability, see Figure 3.18. Therefore, it is not appropriate to directly use the softmax loss function for training a neural network, since it produces features that suffer from this deficiency. To improve this type of neural network, we need to deal with both the intra-class vector distribution and the distribution over the different classes.

Face recognition network trained with center loss

The paper [183] introduced this novel loss function with the aim of suppressing the disadvantages mentioned above. The authors considered how to improve the discriminative power of deeply learned features. The proposed loss function used an idea to minimize the intra-class variation while still ensuring the separability between the features of different classes. This resulted in the following loss function [183]



(a) (b)

Figure 3.18: Deep feature distributions obtained by using (a) training dataset and (b) test dataset [183].

$$L_C = \frac{1}{2}\sum_{i=1}^{m} \|x_i - c_{y_i}\|_2^2 \tag{3.38}$$

where $C_{y_i} \in \mathbb{R}^d$ expresses the y_i^{th} class center of the deep features, the $x_i \in \mathbb{R}^d$ is the i^{th} deep feature belonging to the y_i^{th} class and the m expresses the number of classes.

Considering the loss function, the center of a deep feature should be updated after the deep features have been changed, which is inefficient in practice. Similar to FaceNet, the problem is solved by calculating a new position of the centers within the mini-batch. In addition to this issue, the samples can sometimes be mislabeled, so the scalar α is used to control the learning rate of the centers. Then, the center of the class c_{yi} is updated by [183]

$$\Delta \boldsymbol{c}_j = \frac{\sum_{i=1}^{m} \delta\left(y_i = j\right) \cdot \left(\boldsymbol{c}_j - \boldsymbol{x}_i\right)}{1 + \sum_{i=1}^{m} \delta\left(y_i = j\right)} \tag{3.39}$$

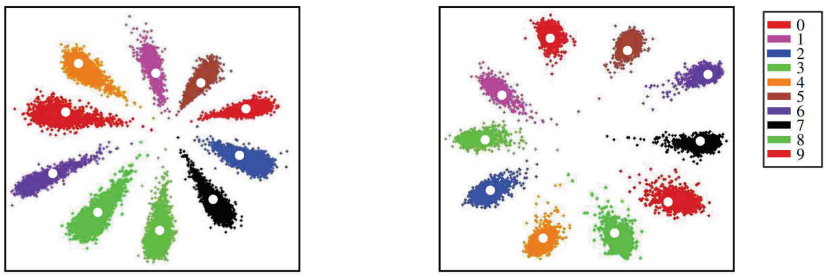
where $\delta(condition)$ is 1 if the condition is satisfied and 0 otherwise.

Finally, the final loss includes the softmax loss and the center loss, which is defined as [183]

$$L_C = L_S + \lambda L_C = -\sum_{i=1}^{m} log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} + \frac{\lambda}{2}\sum_{i=1}^{m} \|x_i - c_{y_i}\|_2^2 \tag{3.40}$$

The deep learning features produced by LeNets trained using the center loss function with different λ scalars are shown in Figure 3.19. Unlike the triplet loss function, the center loss is directly focused on the objective of intra-class compactness, which brings advantages in improving the discrimination of deep features. However, both approaches strive to form the sample pairs or sample triplets from the training dataset, which necessitates the use of optimization methods.

The architecture of this neural network employs the same general concepts described by the previously discussed networks. It consists of convolutional, pooling and fully connected layers, without any special addition. All training samples were preprocessed and cropped to 112×96 RGB images and normalized by subtracting 127.5 and then dividing by 128. The training dataset was collected from the internet (CASIA-WebFace [130], CACD2000 [185], Celebrity + [186]). The authors present results obtained from experiments with three



(a) $\lambda = 0.001$ (b) $\lambda = 0.01$

Figure 3.19: Deep feature distributions for various coefficients λ [183].

different models; the first was trained with softmax loss, the second was trained with center loss only, and the last was supervised by the loss consisting of previous ones. The final accuracy achieved on the LFW dataset is 99.28 %.

3.5.6 SphereFace

The training of the neural network presented above was supervised by the loss functions using the Euclidean distance measure. According to Section 3.5.5, the distribution of the feature vectors corresponds to an angular distribution, but this is not taken into account in the loss function, which uses a distance metric. The *SphereFace* method, published in 2017 [187], takes this finding into account. The essential change in this approach lies in the loss function. Although the loss function used in this approach is similar to the softmax loss, it incorporates an angular margin instead of the Euclidean distance metric. This approach has also been used in other published works, resulting in improved face recognition performance.

Since other algorithms employ similar modifications, we decided to analyze the modifications applied to the softmax loss function. First, we examine the decision criteria of the softmax loss. The best way to illustrate this concept is to use binary class cases, where the posterior probabilities obtained by the softmax loss function are defined as [187]

$$p_1 = \frac{e^{W_1^T x + b_1}}{e^{W_1^T x + b_1} + e^{W_2^T x + b_2}}, p_2 = \frac{e^{W_2^T x + b_2}}{e^{W_1^T x + b_1} + e^{W_2^T x + b_2}} \tag{3.41}$$

where x is the learned feature vector, W_i expresses the weights and b_i is the bias with the bias corresponding to class i. Then, if $p_1 > p_2$, the predicted label is assigned to class 1, and similarly, if $p_2 > p_1$, the label is marked as class 2. From Equation 3.41 it is clear that $W_1^T x + b_1$ and $W_2^T x + b_2$ determine the classification results. Thus, we can claim that the decision boundary is $(W_1 - W_2)x + b_1 - b_2 = 0$. According to [187], the $W_i^T x + b$ can be rewritten as $\left\|W_i^T\right\| \|x\| cos(\theta_i) + b_i$, where θ_i is the angle between x and W_i. When the weight vector is normalized to one ($|W_i| = 1$) and the bias is set to zero ($b_i = 0$), it is worth noting that p_1 and p_2 share the same learned feature vector. As a result, the decision bounds depend solely on θ_1 and θ_2. Overall, the decision boundary becomes $cos(\theta_1) - cos(\theta_2) = 0$. This concept can be extended to the multi-class case. Consequently, the softmax loss function can be modified to [187]

$$L_i = -log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_j e^{W_j^T x_i + b_j}} = -log\frac{e^{\left\|W_{y_i}\right\| \|x_i\| cos(\theta_{y_i,i}) + b_{y_i}}}{\sum_j e^{\|W_j\| \|x_i\| cos(\theta_{j,i}) + b_j}} \tag{3.42}$$

where $\theta_{j,i} (0 \leq \theta_{j,i} \leq \pi)$ is the angle between the vector W_j and x_i. When the weights are normalized by the first norm $\|W_j\| = 1, \forall j$, the function looks like follows [187]

$$L_{ang} = \frac{1}{N} \sum_i -ln \frac{e^{\|x_i\| cos(\theta_{y_i,i})}}{\sum_j e^{\|x_i\| cos(\theta_{j,i})}}. \tag{3.43}$$

Although the modified loss function can be used to monitor the neural network during the training process, it does not necessarily guarantee feature discrimination. Based on the analysis of the softmax loss function, it was observed that the decision boundaries have a strong influence on the feature distribution. This assumption has a significant impact on approaches based on angular metrics. For SphereFace, the loss function was modified by adding a parameter m to control the size of the angular margin.

To illustrate the principle, consider a binary classification task. If class 1 is represented by θ_i, which is given by the angle between a learned feature x and its corresponding weights W_i, then the previously modified loss function requires $cos(\theta_1) > cos(\theta_2)$ to correctly classify x. With the margin parameters, the equation can be changed to $cos(m\theta_1) > cos(\theta_2)$. Assuming that $m > 2$ is an integer, this decision is stricter than the previous one, because the angle must be smaller to get the same value as in the previous case. This also applies from a class 2 perspective, $cos(m\theta_2) > cos(\theta_1)$. So for class 1, the boundary is $cos(m\theta_1) = cos(\theta_2)$ and for class 2, the boundary is $cos(m\theta_2) = cos(\theta_1)$. Assuming that all training samples are correctly classified, the decision boundaries of all training samples produce the angular margin $\frac{m-1}{m+1}\theta_{\frac{1}{2}}$, where $\theta_{\frac{1}{2}}$ expresses the angle between W_1 and W_2. The modified loss function is defined as [187]

$$L_{ang} = \frac{1}{N} \sum_i -log \frac{e^{\|x_i\| cos(m\theta_{y_i,i})}}{e^{\|x_i\| cos(m\theta_{y_i,i})} + \sum_{j \neq y_i} e^{\|x_i\| cos(\theta_{j,i})}} \tag{3.44}$$

where $\theta_{y_i,i}$ is the angle in the range of $[0, \frac{\pi}{m}]$. This loss function is called *A-Softmax* [187] and it is based on the SphereFace algorithm.

Although the changes are feasible in theory, the loss is difficult to optimize in CNNs. To solve this issue, the authors proposed a monotonically decreasing angular function $\psi(\theta_{y_i,i})$, which should be equal to $cos(\theta_{y_i,i})$ in the range $[0, \frac{\pi}{m}]$. Therefore, the final A-Softmax loss function was partially changed and is formulated as [187]

$$L_{ang} = \frac{1}{N} \sum_i -log \frac{e^{\|x_i\| \psi(m\theta_{y_i,i})}}{e^{\|x_i\| \psi(m\theta_{y_i,i})} + \sum_{j \neq y_i}^{n} e^{\|x_i\| cos(\theta_{j,i})}} \tag{3.45}$$

where the $\psi(m\theta_{y_i,i}) = (-1)^k cos(m\theta_{y_i,i}) - 2k$, $\theta_{y_i} \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$. The integer m is intended to control the size of the angular margin.

Similar to previous cases, the authors employed a CNN that includes the convolutional layer, the pooling layer, and the fully connected layer, while he neural network was trained on CASIA-WebFace and tested on LFW (Section 3.2.3) and YTF (Section 3.2.3). To find out the influence of the parameter m on the accuracy of the neural network, the authors performed experiments with different values. The results showed that m has a significant influence on the accuracy, see Figure 3.2. The distribution of the learned features is visualized in Figure 3.20. Overall, SphereFace achieves 99.42 % accuracy on the LFW dataset.

SphereFace was one of the pioneering approaches that used the concept of angles instead of a distance metric. However, the A-Softmax loss function contains approximations that can affect the stability of neural network training. Ensuring stability during training remains

Table 3.2: Influence of sotfmax's m parameters on accuracy.

Dataset	**Original**	$m = 1$	$m = 2$	$m = 3$	$m = 4$
LFW [%]	97.88	97.90	98.40	99.25	99.42
YTF [%]	93.10	93.20	93.80	94.40	95.00

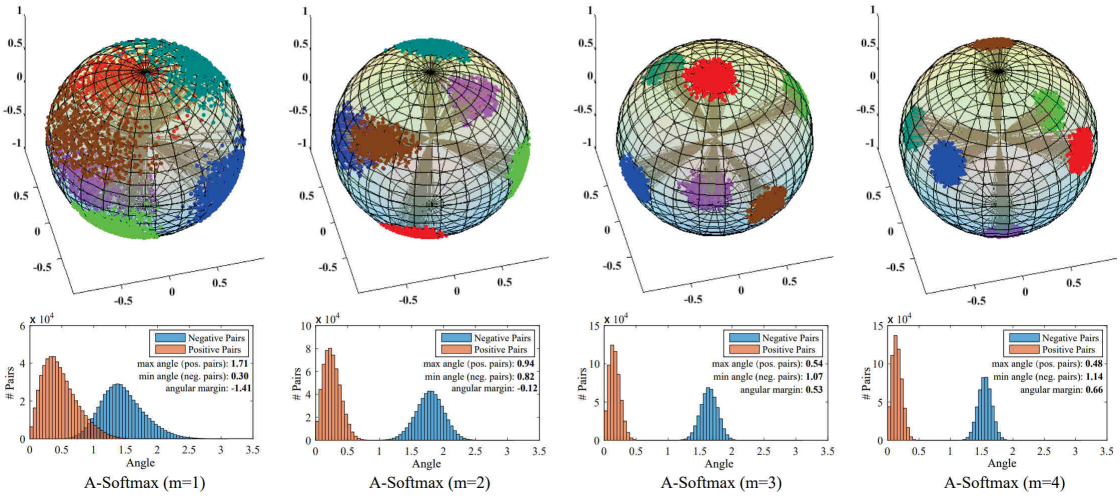


Figure 3.20: Visualization of learned features dependent on m parameters [187].

a critical goal for other state-of-the-art approaches that aim to improve the usability and performance of the loss function.

3.5.7 CosFace

Similar to previous approaches, the *CosFace* loss function, as described in [188], is based on the concept of angles. Through an analysis of the SphereFace approach, its limitations have been identified. Hence, the primary objective of this approach, known as CosFace, is to refine the approximations employed in the SphereFace loss function. In addition, this approach aims to improve the accuracy of face recognition. One of the shortcomings of the A-Softmax loss is the non-monotonic nature of the cosine function, which SphereFace mitigates through function approximations.

During the development of the CosFace loss function, it was found that the face recognition score of a test pair is usually given by the cosine similarity between the two future vectors. To achieve this, the previously defined softmax function without bias is adapted, where $W_j^T x$ is replaced by $|W_j|\,|x|\,cos(\theta_j)$, with θ_j representing the angle between W_j and x. According to A-Softmax, W is modified by L2 normalization to $\|W_j\| = 1$. Consequently, we assume that the norm of the feature vector x does not contribute to the scoring function, so the norm is fixed to $\|x\| = s$. Thus, the posterior probability is only given by the cosine of the angle. The modified formula is thus defined as [188]

$$L_{lmc} = \frac{1}{N} \sum_i -log \frac{e^{\|x_i\| \psi(m\theta_{y_i,i})}}{e^{\|x_i\| \psi(m\theta_{y_i,i})} + \sum_{j \neq y_i}^{n} e^{\|x_i\| cos(\theta_{j,i})}} \tag{3.46}$$

The authors refer to this formula as the *Normalized Version of the Softmax Loss* (NSL) [188]. With the loss function, the neural network produces separable feature vectors in

a radial distribution. However, the feature vectors are not discriminative enough. Therefore, the authors introduce a modified loss function by adding a cosine margin to the classification boundary.

Taking the binary class example from SphereFace, if the angle θ_i is given by the learned feature vector (x) and the corresponding weight vector W_i, then it is necessary to satisfy $cos(\theta_1) > cos(\theta_2)$ to correctly classify x_1. Since the margin between two classes enforces the discrimination between them, the comparison equation is modified by adding the margin parameter m. Unlike the A-Softmax, the deep feature discrimination is enforced by the cosine margin instead of the angle margin.

In A-Softmax, the angle is multiplied by the margin parameter, while in CosFace the loss is modified by changing $cos(\theta)$. For the classification feature vector fall into class 1, it must satisfy the following condition $(cos(\theta_1)-m) > cos(\theta_2)$ and for class 2 $(cos(\theta_2)-m) > cos(\theta_1)$ similarity. Assume that $m \geq 0$ is a fixed parameter introduced to control the magnitude of the cosine margin. From the point of view of class discrimination, the $cos(\theta_i) - m$ is lower than $cos(\theta_i)$, so this ensures a stronger classification. After generalizing this example to multiclass, the *Large Marin Cosine Loss* (LMCL) is formulated as follows [188]

$$L_{lmc} = \frac{1}{N}\sum_{i} -log\frac{e^{s(cos(\theta_{y_i,i})-m)}}{e^{s(cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s(cos(\theta_{j,i})-m)}} \tag{3.47}$$

$$cos(\theta_{j,i}) = W_j^T x_i, W = \frac{W^*}{\|W^*\|}, x = \frac{x^*}{\|x^*\|} \tag{3.48}$$

where N is the number of training samples, x_i is the vector of the i^{th} depth features, W_j is the column of the weight matrix for the j^{th} class, and θ_j is the angle between W_j and x_i.

Let us look at the normalization of the weight vector and the feature vector. From [188], it is clear that the weight vector can be normalized by L2 norm, which is used in the A-Softmax loss function. However, if the feature vector is not normalized, the softmax loss implicitly learns the L2 norm of the feature vector and the cosine value of the angle. Therefore, the LMCL uses both normalization and, for the weights, a scaling parameter that controls the magnitude of the radius.

Using normalization, the CosFace algorithm with LMCL achieved an accuracy of 99.33 % on the LFW dataset and 96.1 % on the YTF dataset.

3.5.8 ArcFace

In 2018, a new method was introduced that leverages the radial distribution of features to achieve superior performance, see Section 3.5.5. To train the new neural network, called *ArcFace* [189], the softmax loss function was also modified. What exactly is an ArcFace Loss Function? In simple terms, it is an approach that further modifies the loss function with respect to angle distribution.

The main goal is to further improve feature discrimination while maintaining separability, which should be enforced by changes in decision boundaries. As we already know, the margin of A-Softmax (SphereFace) is enforced by multiplying the angle before calculating the cosine (multiplicative angular margin), while CosFace determines the margin by subtracting the margin value (additive cosine margin) from the calculated cosine value. In the case of SphereFace, a significant disadvantage arises due to the approximation used to

calculate the monotonic function. This factors have been taken into account in the design of the ArcFace.

In ArcFace, the modification of the softmax function lies in the following changes. For simplicity, the bias is set to $b_j = 0$ [187]. Furthermore, this results in $W_j^T x_i = \|W_j\| \|x_i\| cos(\theta_j)$, where θ_j is the angle between the weight W_j and the feature x_i. Both the weight and feature vectors should be normalized by the L2 norm, then $W_j = 1$ and $x_i = 1$ [190]. In the loss function, the weight is scaled by s to determine the magnitude of the vector. The normalization and the weights cause the prediction to depend only on the angle θ between the weight and the feature vector. Accordingly, the loss function for i^{th} class is defined as [189]

$$L_{part} = -\log \frac{e^{s\cos(\theta_{y_i})}}{e^{s\cos(\theta_{y_i})} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos(\theta_j)}} \tag{3.49}$$

This function produces embedding vectors that are distributed around feature center on a hypersphere. To enforce better intra-class compactness and inter-class discrepancy, the loss function added angular penalty. Finally, the loss function for i^{th} class is defined as follows [189]

$$L_{part} = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos(\theta_j)}} \tag{3.50}$$

The authors used a proven experiment with 2D vector to investigate the feature distribution, which shows that the ArcFace loss produces a wider gap between the nearest classes than the softmax loss. In contrast, the Softmax produces noticeable ambiguity in the decision boundaries. From a numerical point of view, A-Softmax, CosFace and ArcFace enforce the compactness of the intra-class features. The second aspect of investigation is the geometric difference, where the focus is on the geometric distance.

The ArcFace has a constant linear angular margin throughout the interval. While, the other loss functions have a non-linear angular margin, see Figure 3.21. Research has shown that small changes in the angular margin can affect the training process.

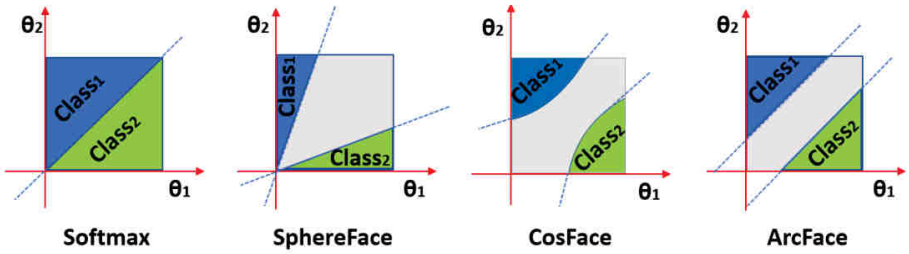


Figure 3.21: Decision margins of different loss functions where the dashed lined represents the decision boundary, whereas the gray color marks the decision margins.

The ArcFace has surpassed the state-of-the-art in face recognition, achieving a remarkable accuracy of 99.53 % on the LFW. The next accuracy evaluation was performed on the CFP-FP dataset, where ArcFace achieved 95.56 % accuracy. It is obvious that the algorithm is capable of successfully solving the face recognition task.

3.5.9 MagFace

Due to the performance of the ArcFace algorithm, researchers have focused on further improving this algorithm. In CosFace, ArcFace employs the angle of the distribution of the vectors to ensure minimizing the intra-class feature distribution and maximizing the distance between clusters of identities. However, the authors of these algorithms did not take into account the magnitude of the feature vector.

In [191] an experiment was presented to prove the influence of vector magnitude on face recognition performance. This leads to the conclusion that the size of the vector magnitude correlates with the quality of a face image, see Figure 3.22.

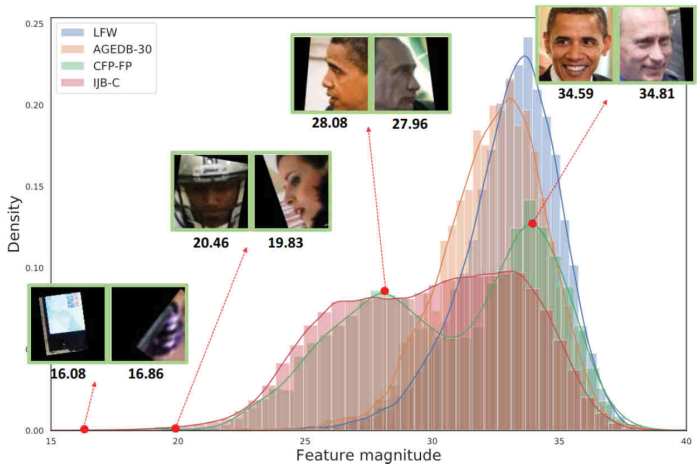


Figure 3.22: Distribution of magnitudes across different datasets [191].

Based on this finding, the ArcFace loss function was modified to utilize the vector magnitude, which is defined as follows [191]

$$L_{Mag} = \frac{1}{N}\sum_{i} -\log \frac{e^{s\cos\left(\theta_{y_i}+m(a_i)\right)}}{e^{s\cos\left(\theta_{y_i}+m(a_i)\right)} + \sum_{j=1,j\neq y_i}^{N} e^{s\cos\theta_j}} + \lambda_g g(a_i) \tag{3.51}$$

where λ_g controls the trade-off between classification and regularization losses. The magnitude of the feature vector a_i is bounded in $[l_a, u_a]$. The $g(a_i)$ and $m(a_i)$ are strictly decreasing convex and increasing convex functions, respectively. The loss function always has the following properties during its optimization

- Property of convergence - *for $a_i \in [l_a, u_a]$, L_i is a strictly convex function which has a unique optimal solution a_i^*.* [191]
- Property of monotonicity - *the optimal a_i^* is monotonically increasing as the cosine-distance to its class center decreases and the cos-distances to other classes increase.* [191]

For the experiments, the authors used $m(a_i)$ as a linear function defined on $[l_a, u_a]$ with $m(l_a) = l_m, m(u_a) = u_m$ by the following equation [191]

$$m(a_i) = \frac{u_m - l_m}{u_a - l_a}(a_i - l_a) + l_m \tag{3.52}$$

where l_m, u_m denote hyperparameters and $g(a_i)$ is given by [191]

$$g(a_i) = \frac{1}{a_i} + \frac{1}{u_a^2} a_i \tag{3.53}$$

This allows the ambiguous samples to be pushed away from the class centers and then pulled back toward the origin. To achieve this improvement, the original loss function was modified by replacing the edge factor with an adaptive edge. Unlike the feature distribution produced by the ArcFace loss, the MagFace loss distributes the vectors according to their direction and magnitude. The MS1M-V2 [190] was used for training due to the small number of noisy face images compared to the MS1M [126].

Using the MagFace algorithm, the accuracy on the LFW dataset was further improved to 99.83 %.

3.5.10 QMagFace

In the above face recognition method, the magnitude of the vector is utilized during the training process. Moreover, this concept of feature vector size has been extended to the comparison stage, giving rise to a novel metric known as *Quality-Aware Comparison Scoring* [192] has been proposed. The training of the neural network is enforced by the MagFace loss function, see Eq. 3.51. The strength of Quality-Aware Face Recognition is based on a comparison metric defined as

$$\hat{s}\,(s, q_1, q_2) = \omega(s) * \min\{q_1, q_2\} + s \tag{3.54}$$

where s denotes the standard comparison score $s = cos(e_1, e_2)$. q_1, q_2 express the magnitudes, and $\omega(s)$ is a quality-weighting function, defined as follows

$$\omega(s) = \min\{0, \beta * s - \alpha\} \tag{3.55}$$

where β and α are trainable parameters.

Currently, the accuracy of face recognition algorithms is generally above 99.70 %. The QMagFace-100 (with IResnet100 [193] feature extractor) achieved an accuracy of 99.83 % on the LFW dataset. QMagFace outperforms accuracy on the CFP-FP and AgeDB datasets, although other algorithms outperform it on the LFW.

3.5.11 Summary

The most significant improvement in face recognition performance came from the use of neural networks. In recent years, progress in individual neural networks has not been substantial. However, there has been a noticeable shift due to changes in loss functions that lead to better exploitation of the angular distribution. Since the potential for improving face recognition algorithms is limited, the significance of these changes is often obscured by preprocessing algorithms applied to training samples, such as padding of the detected face.

3.6 Devices for neural network acceleration

Neural Processor Unit (NPU) [194] is a type of computing unit designed to perform tasks that are domain-specific in nature. As a result, these chips are generally not based on the Von Neumann architecture. One of the goals of the NPU is to reduce power consumption

while maintaining maximum computational throughput. For example, Google developers have developed a chip called the *Tensor Processor Unit* (TPU) [150]. The TPU is designed to process large amounts of data with low precision. However, neural networks consist of parameters represented by real values (floating-point numbers), which are less efficient to process than integers. Therefore, before the neural network is processed by an NPU, the model parameters are mapped to another representation using quantization algorithms, usually with low precision.

According to [195], quantization techniques are used by virtually all NPU chips. Although quantization reduces the accuracy of neural networks, the gain in acceleration overcomes this disadvantage in many applications. By analyzing the CNN, we can see that most common operation is a convolution. Therefore, the NPU contains a unit that is primarily designed to perform matrix multiplication, with an emphasis on speed and efficient operation.

Note that most NPU chips are proprietary and descriptions of their technologies are not published. The following section describes the known technologies associated with the NPU.

3.6.1 Quantization

As mentioned above, the quantization process in neural networks involves mapping the floating point value, commonly represented by the IEEE 754 [196] standard, to compact representations according to well-defined rules. For example, on the hardware side, the processing of integer values is easier than processing floating-point values. The parameters of the neural network can be quantized due to the observation that neural networks are over-parameterized [197] and robust to noise [198]. Therefore, there is an opportunity to reduce the number of bits per parameter with minimal impact on accuracy. Except this, the parameters of the over-parameterized network can be reduced by pruning [197] or knowledge distillation. Even when processing neural networks on Graphics Processor Unit (GPU), there is an effort to reduce floats to a format with less accuracy, i.e. *float16*. Except for special cases, the parameters of a neural network are represented in double data type. Let us see how the basic concept of quantization is defined.

When training the neural network, the primary goal is to minimize the loss function. Through quantization, the goal is to reduce the precision of the parameters in the neural network and the corresponding activation maps with minimal impact on the generalization accuracy of the model to achieve faster inference. The following subsections describe the basic types of quantization methods and their properties [195].

Uniform quantization

First, we need to define a function that maps a value to another representation. The easiest way is to use the function where the input is represented by a real value and its map to a lower precision range. We assume that the distance between the quantized values is preserved; this method is called *uniform normalization* [195]. Theoretically it is defined as [195]

$$Q(r) = \mathrm{Int}(r/S) - Z \tag{3.56}$$

where Q expresses the quantization operator, r is a real value input, S is the scaling factor and Z is the integer zero point. The de-quantization function is defined as [195]

$$\tilde{r} = S(Q(r) + Z). \tag{3.57}$$

where $\tilde{r}$ is the recovered real value.

Symmetric and asymmetric quantization

Quantization maps a value to a well-defined range that depends on the scaling factor S. We can design the quantization method to produce values in the symmetric or asymmetric range. This range is defined by two parameters called α and β, where the first defines the minimum value and the second defines the maximum of the range, then S is defined as [195]

$$S = \frac{\beta - \alpha}{2^b - 1}. \tag{3.58}$$

The process of selecting the parameters is referred to as *calibration*. If the parameters satisfy $-\alpha = \beta$, the quantization is called symmetric. Otherwise, it is considered asymmetric quantization. By modifying Eq. 3.56, the symmetric quantization ($Z = 0$) is defined by [195]

$$Q(r) = \text{Int}\left(\frac{r}{S}\right). \tag{3.59}$$

The disadvantages of symmetric quantization become apparent when the range of values is non-symmetric and skewed.

Dynamic and static quantization

Next, we need to decide when the values will be quantized. In neural networks, the weights are defined during model training and their range of values is known in advance [195]. On the other hand, the activation outputs should produce values in a variable range depending on an input sample. In this case, the predefined range is not suitable. Fortunately, besides static quantization, there is also dynamic quantization, which calculates a quantization coefficient for each activation map. However, this approach has more overhead, requiring minimum, maximum, and percentile calculations.

When we use static quantization, the range is determined before the runtime. It can be used when we have deep knowledge about the values. During the runtime, the accuracy of neural networks can be negatively affected by static quantization. Nonetheless, this type of quantization is often used due to latency requirements.

Quantization granularity

The quantization methods can be divided into categories according to how the clipping range is calculated for the neural network weights. These categories include [195]:

- *Layerwise quantization* - in convolutional filters, the clipping range is calculated for all weights in the layer. However, in convolutional filters, the variance of the weights within the layer can vary widely. Thus, a convolutional kernel with a relatively narrow range of parameters can be negatively affected by this type of quantization if another kernel has parameters in a wider range.

- *Groupwise quantization* - using this approach, quantization can be performed on a group of different channels within a single layer. In this way, the disadvantages caused by varying parameters can be suppressed.
- *Channelwise quantization* - with this option, the clipping range for convolution filters is defined separately within channels. This method provides better quantization resolution and often results in higher accuracy.
- *Sub-channelwise quantization* - compared to the previous method, sub-channel quantization ensures different clipping ranges of convolution kernels within a channel. However, the approach introduces significant overhead.

Fine-tuning methods

In the text above, we described different approaches that can be used for quantization. However, when applying quantization to neural networks, it is often necessary to tune their parameters. We can tune the parameters of quantized neural networks either through *Quantization-Aware Training* (QAT) [195] or through a method that does not involve retraining, called *Post-Training Quantization* (PTQ) [195].

Using QAT, we aim to optimize the quantization coefficients. Each training iteration consists of a forward and a backward pass, during which the gradient calculation is performed on the quantized model in floating point. The model parameters are updated after each gradient update and are then quantized again.

The rounding operator in Eq. 3.56 can cause the gradient of this operator to be zero almost everywhere. Thus, the question is how to edit the weights in the neural network according to the quantization parameters when the gradients during training have no relation to the quantization parameters. This can be solved by the *Straight Throughout Estimator* (STE) [195], which is essentially based on replacing the rounding operation with an identity function. In the same way, the other approaches can be used to solve the drawback.

Another fine-tuning method, PTQ, is based on the idea of performing quantization on the trained model. The trained model is analyzed and its parameters are quantized without retraining it. In contrast to QAT, this approach may result in lower accuracy of the quantized model.

The final method is *Zero-Shot Quantization* (ZSQ) [195], which addresses the issue of insufficient data for calibration. This method uses a subset of the training data to calculate the scaling factor and the boundaries of the clips. In addition, the training subset is used to fine-tune the model, allowing adjustments to the model parameters and restoring accuracy that may have been lost. However, data availability can sometimes be limited, and ZSQ provides a potential solution to this challenge.

In [199], two different levels of zero-shot quantization were presented:

- Level 1 - no data and no fine-tuning (ZSQ + PTQ)
- Level 2 - no data but requires fine-tuning (ZSQ + QAT)

Level 1 is for fast quantization without fine-tuning. On the other hand, in level 2, the fine-tuning helps the quantized model to recover from the loss of accuracy.

Overall, the above information on quantization properties and approaches is merely a brief summary. In practice, quantization is a complex process involving other algorithms to optimize a neural network model. Authors in [195] present techniques related to sub-INT8

quantization. The low-bit data format allows the operations to be performed efficiently on the hardware device. Hardware accelerators for neural networks are designed to handle large integers. The smaller the range, the more operations per second should be processed.

3.6.2 Matrix multiplication

Modern AI accelerators are designed to compute a large number of operations. From a high-level architecture perspective, the accelerator divides the operations into parallel computing and memory units organized in a two-dimensional structure to support common matrix-vector multiplications. According to the [200], the most common operations in the DNN are matrix multiplication and convolution, which cover over 90 % of the operations in a neural network. Mathematically, the computational patterns of matrix multiplication and convolution are different. However, these operations can be cross-transformed by remapping. The convolution operation can be transformed into matrix multiplication using a *Toeplitz matrix* [200]. The AI accelerators usually use only one such operation, so it must include the unit or algorithm to remap the rest of the operation. One way to ensure matrix multiplication in the accelerators is to use the systolic array [200].

The premise of the systolic array is based on replacing the single processor with an array of regular processing elements. While the Single Instructions Multiple Data (SIMD) [200] approach performs the same operation on multiple data, the arithmetic element in the systolic array can perform different operations. For matrix multiplication, the systolic array is arranged on the 2D grid where each of the elements incrementally accumulates the values of the products. Sometimes the units are called *multiply-accumulate*, often abbreviated as MAC [201].

3.6.3 Available devices

From the available NPU devices for accelerating neural networks, we have selected several representatives that can be used for accelerating face recognition algorithms. First of all, the pioneer of such devices is Google, which has developed the Google Tensor Processing Unit.

Google Tensor Processing Unit

The rapid development in the field of neural networks forced Google to develop a chip capable of both training and inferencing neural networks. In 2015, the first *Tensor Processing Units* (TPUs) [150] were used internally by Google. The TPU is designed to process large amounts of data with less precision, specifically low-precision integers.

Architecturally, the workhorse of the TPU is based on the 8-bit systolic matrix to provide matrix multiplication [202], see Figure 3.23. In this case, the units of the multiplier are called MACs (Multiply-Accumulate). The Google Edge TPU only supports 8-bit arithmetic. Therefore, a neural network model must be quantized before it can be used with the TPU. The matrix unit generates a partial sum of 256 elements per clock cycle. The weights for the multipliers are stored in a weight FIFO that reads data from an external DRAM memory, the weight memory. In addition, the intermediate results are stored in the 24 MiB Unified Buffer, which provides the data to the matrix multiplier.

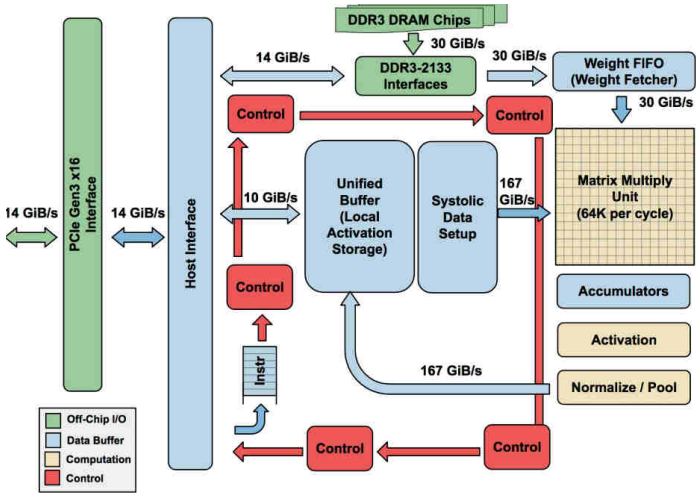


Figure 3.23: Architecture of the first generation of TPUs [200].

ARM – Machine Learning Processor

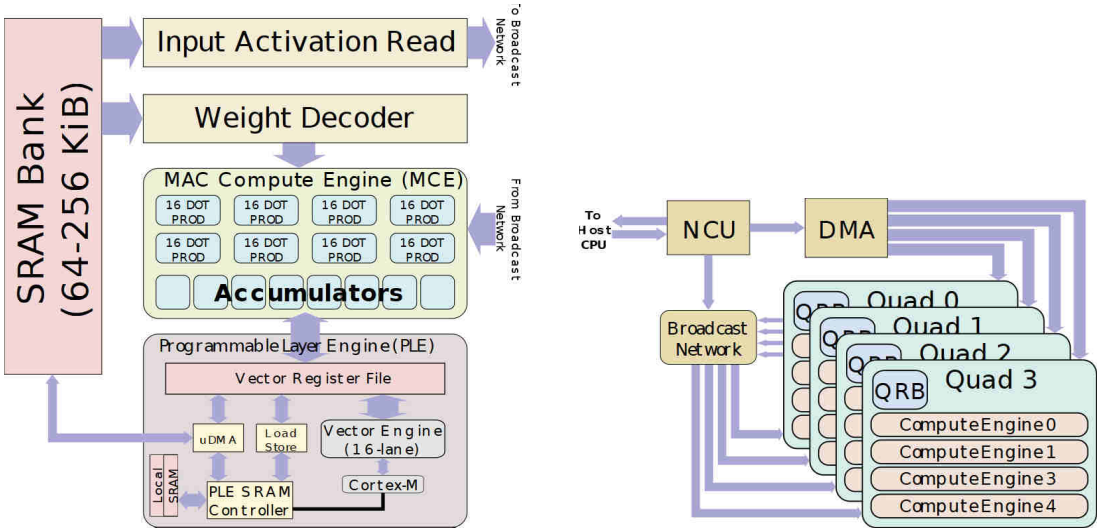
The Machine Learning Processor (MLP) [201] was introduced by the ARM consortium in 2019 as a processor architecture. The primary focus of the design, as envisioned by the authors, is scalability. This means that the architecture can be customized and synthesized for specific use cases, with a performance range from 1 TOPS to 10 TOPS. In terms of power consumption, the choice of a smaller model and a chip with lower throughput is the more favorable approach.

From a high-level perspective, the architecture (Figure 3.24) consists of DMA, which is responsible for communication between the memory and the processor [201]. To better optimize the computational process, the DMA engine uses information about neural network layouts that are predefined during the NPU design process. Next, there is a component called the Network Control Unit, which is responsible for communication. Finally, the most important part is the *Compute Engine* (CE) [201], which is the core of the processor. As mentioned earlier, the primary function of the NPU is to perform fast convolution calculations. Within the NPU, there exists a specialized unit known as the *MAC Compute Engine* [201], which is specifically designed to perform fixed-function multiply-accumulate operations. Each individual MAC unit within this engine is capable of computing 16 8-bit dot product operations.

In addition to convolution operations, the neural network model includes other types of operations, such as normalization, addition, and pooling layers. To compute the various operations, the CE uses the *Programmable Layer Engine* (PLE) [201], which is a programmable processor that supports vector operations and is designed to perform post-processing operations and custom functions. The MCE and PLE utilize the slice of SRAM.

In terms of data processing, the MLP is statically scheduled, which means that all operations related to the hardware architecture are planned during compilation. During compilation, the compiler ensures that the neural network model is interleaved over the sources. The minimum memory of MLP is SRAM, which is used to store input feature maps, model weights, and output feature maps. The compiler allocates the space for them in CE.

Now let us see how the code is executed in the CE. First, each MCE reads 2D patches of the input map from its local SRAM and then sends them to the broadcast neural network.



(a) (b)

Figure 3.24: Architecture of ARM MLP [201].

Then, the broadcast neural network maps the 2D patches into 3D blocks of input activations and sends them to all MCEs in the MPLE. At the same time, each MCE loads weights from SRAM and decompresses them. Then, the input feature map is computed by the MCE and the results for the output feature map are stored in 32-bit accumulators. Finally, the value is mapped to 8-bit and shifted to PLE. When the values arrive in the register file of the PLE, the Arm Cortex-M CPU runs the vector engine to perform the appropriate operation on the register file.

Intel Movidius Myriad X Vision Processing Units

The Intel Movidius Myriad X Vision Processing Units (MA2085) [203] are the third generation of dedicated hardware accelerators for deep neural networks. The core of the chip is based on a Vision Processing Unit (VPU) architecture that includes the Neural Compute Engine, Vision Accelerators, Imaging Accelerators, CPUs, and 16 SHAVE processors. The SHAVE processors [204] are based on the 128-bit VLIW[14] architecture, which allows multiple application pipelines to run simultaneously.

In terms of interfaces, the processor supports MIPI lines, USB 3.1, Quad SPI, I2S, PCIe 3.0 and 10 GbE.

Before this chip, an older version was introduced. The chip is called MA2450 and was introduced in 2016. However, its production was discontinued in 2019.

To make this easy to use, the chips are manufactured in a form of a USB stick. The first device, called Intel Neural Compute Stick, was based on MA2450, while the second generation used MA2085 chip.

The OpenVINO[15] library is used to operate the device. Unfortunately, the last version that supports the accelerators is OpenVino 2022.3.

[14] Very long instruction word
[15] https://www.openvino.com

Hailo

The year 2022 saw the emergence of a new type of accelerator called Hailo 8[16]. This accelerator, as stated in their PR material, has the ability to significantly accelerate neural network response calculations for input data. While the company has provided limited information about its architecture, it is understood that the neural network undergoes quantization and is then mapped onto a resource graph. Our observations further indicate that this quantization method specifically utilizes PTQ. In the subsequent phases, the graph is mapped onto the chip's physical resources, with a strong focus on optimizing performance and efficiency.

Intrigued by the speed potential of this accelerator, we decided to conduct an evaluation of its suitability for face recognition tasks. The company offers the accelerator in various forms, including a standalone chip and compute modules compatible with mPCIE and M.2 interfaces.

Rockchip NPU

The Rockchip NPU[17] has been developed by Rockchip, a Chinese semiconductor company that is involved in the processor manufacturing. This NPU is designed to be integrated primarily into their processors.

The biggest challenge is documentation. Although information about the architecture of the Rockchip NPU is not readily available, we have discovered that it supports INT4, INT8, INT16 and FP16 hybrid operations [205]. By studying the SDK documentation, we found that the model can be quantized using asymmetric quantization. In addition, the quantization process can be controlled by the optimization level parameters selected by the user during initialization.

[16] https://hailo.ai/products/hailo-8/
[17] https://github.com/rockchip-linux/rknpu2

Chapter 4

Experiments with face detection and recognition algorithms

One of the primary objectives of the thesis is to study face detection and recognition algorithms applied to face images captured from different angles. Due to the limited availability of such datasets, a dataset generator was designed and developed. The idea of using a generated dataset stems from both the need for specialized data samples and privacy concerns. By using image processing algorithms, simulating optical properties, and projecting 3D head models onto a plane, we gain the ability to generate a dataset containing specific images that may be more difficult to obtain in larger quantities in a real-world environment. Such images can be used to augment existing datasets or to create new datasets to evaluate available algorithms. Several experiments are conducted to validate the capabilities of the generator and the resulting dataset. The first experiment focuses on assessing the similarity between the generated dataset and a real face dataset. Additionally, the generated dataset is subjected to tests using state-of-the-art face detection and recognition algorithms.

In recent years, there have been advances in methods for generating and augmenting face data, many of which rely on generative neural networks. With these types of networks, new datasets can be created either directly, or by using existing face images. However, generative neural networks may not always faithfully preserve facial features, and we wondered what such a distortions look like. To address this question, an experiment was conducted in which a frontal photo was transformed into a profile using artificial intelligence.

4.1 Our approach to obtain datasets for face recognition

Datasets serve as the foundation for training and testing machine learning algorithms. These datasets typically consist of face images, which can be acquired through three primary methods. The most straightforward approach involves capturing images directly with a camera. Alternatively, computer vision and machine learning techniques can be employed to generate these images. The generation methods encompass the utilization of neural networks or the utilization of 3D scanned head models.

The research objective is to determine how face recognition performance is affected by the rotation of the face. This evaluation requires a dataset with a substantial number of subjects. Nevertheless, the availability of such a dataset is limited. In [136], a dataset of faces captured by a camera from different angles was presented. However, this dataset has fixed environmental factors, and its extension is practically impossible due to the use of a special

camera setup for the capturing. Another method to generate face images in specific poses involves the use of generative neural network models [206].

We decided to develop an approach that preserves facial characteristics and simulates real conditions using 3D scanned head models with facial texture instead of fully generated images. Consequently, we introduce our novel method for acquiring a face dataset from 3D head scans.

4.1.1 HeadViewer and creation of a custom face dataset

In 2015, we conducted initial experiments with the aim to find out how head rotation in the pitch axis affects the detection and recognition performance. To do this, we used an aerial work platform (AWP) to obtain the real images with the head rotated in the pitch and yaw axis. Although the dataset, namely Airfield Face 2015[1], is composed of only three subjects (see Figure 4.1), it was sufficient for the first experiments. In order to capture the head (face) at different pitch, we defined two scenarios, as shown in Figure 4.2. The dataset is mainly focused on evaluating the influence of pitch rotation on face detection. In addition to the frontal images, the face images were rotated in two othe yaw angles.

Figure 4.1: Example of images from our Airfield Face 2015 dataset.

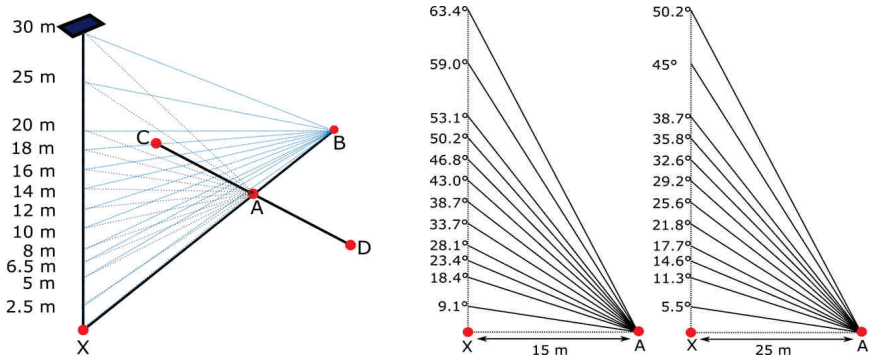


Figure 4.2: (a) Sketch of the capturing scene using the AWP, where A, B, C, D represent spots where the individuals stood and X denotes the AWP position. Distances: $|XA| = 15\ m$, $|XA| = 25\ m$, $|AD| = 10\ m$ and $|CA| = 10\ m$. (b) Angles calculated based on the distances.

The results were used to develop software to simulate cases where face detection does not work as well as it should. We designed the software, called HeadViewer [TG.3], primarily as a tool for installing security cameras, see Figure 4.3. The setup of the cameras has

[1] internal dataset

a direct impact on the utilization of the video captured by the cameras in computer vision algorithms [TG.4]. The software made a significant contribution to security in 2015 and 2016, when it was used by police and companies working with security cameras. However, due to the rapid development of new technologies, HeadViewer has become obsolete from today's point of view.

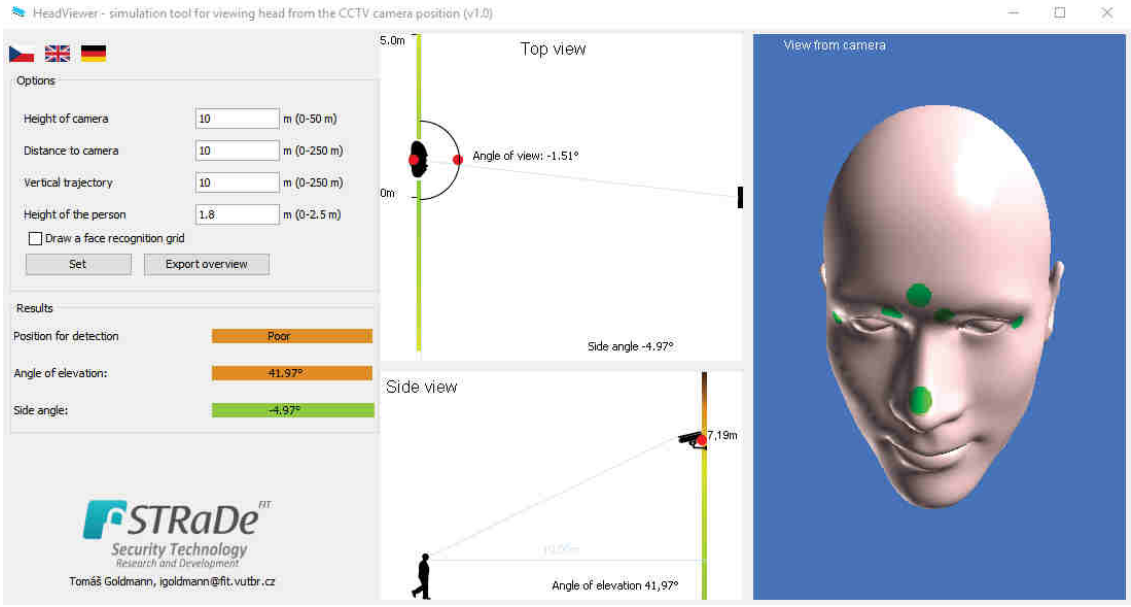


Figure 4.3: HeadViewer screenshot.

4.1.2 New sensors for capturing 3D models of faces

While developing HeadViewer, we encountered a lack of rotated face images. Capturing the face in a specific pose is difficult due to the need to align the face and ensure the cooperation of the captured subjects. Although the neural network can be employed to generate the face images in a specific position, such images do not correspond to real people and may suffer from distortion of certain characteristics. To prove these claims, we have conducted experiments with a neural network capable of generating a face in different positions [207][208] derived from a frontal image. We found that the generated profile image has a different shape (especially around the nose (1)) than the real face profile image, see Figure 4.4. Likewise, the angle between the lines (2) and (3) is different; it is smaller in the AI-generated image than between the lines in the real image.

Other approaches to obtain the accurately positioned face images have emerged from the use of the 3D head model. Images for the dataset are generated by projecting the 3D head model onto the 2D plane using computer vision methods. The first step in the process is to obtain a 3D head model. As part of police cooperation, we are asked to build a customizable 3D head scanning device. Such 3D scanners are usually based on Time of Flight (ToF) [209], structured light [209], and photogrammetry [210]. After consulting with the police and a specialist in anthropometry, we decided to use the 3D head models of real people to generate a dataset of 2D images.

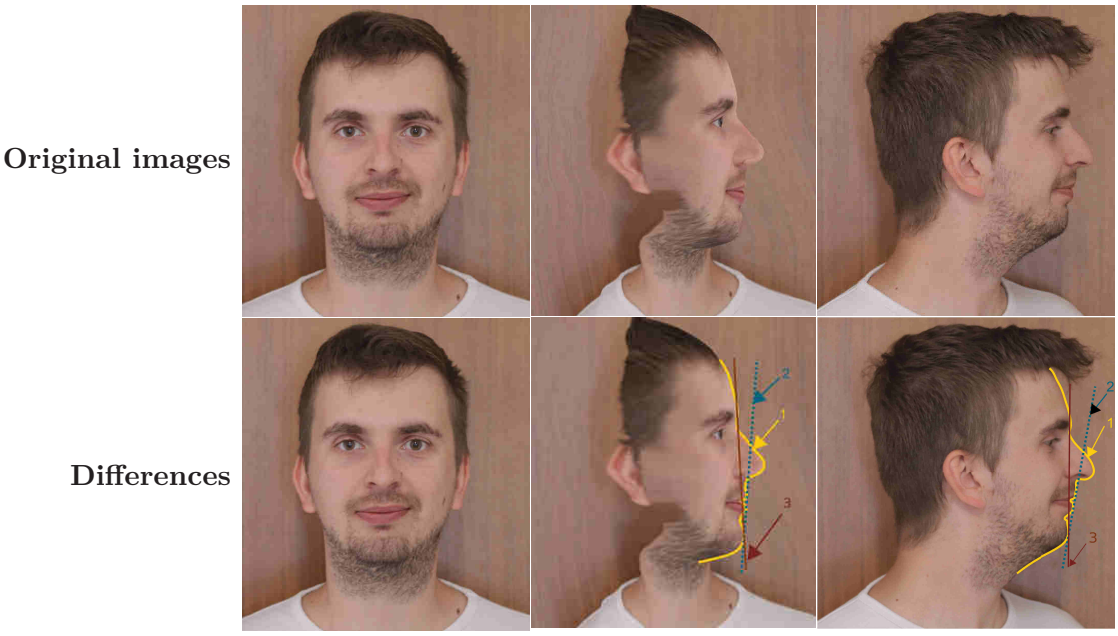


Figure 4.4: Experiments with the AI generator of face images in different positions: (a) reference image, (b) generated frontal image and (c) real frontal image.

For our face profile scanning sensor [TG.5], we chose to use photogrammetry. Unlike similar commercial products Vectra H1[2] and Vectra H2 [3], our device can be modified into a 3D face recognition sensor or fit behind a semi-transparent mirror.

Sensor design and construction

Based on our experience, we decided to use cameras with frame buffers. This allows all the images to be captured in a single moment, with a minimal time difference between captures. Another consideration was the number of cameras needed to capture the 3D face profile. In total, we decided to place the six cameras at the vertices of the pentagon shape, see figure 4.5. Due to the risk of face spoofing when using an unattended sensor, the prototype was equipped with a thermal sensor for liveness face detection (LFD).

From a technical point of view, the device consists of six cameras, each capable of capturing images with a maximum resolution of 3,840×2,160 pixels. These cameras are equipped with lenses with a focal length of 4.3 mm and a diagonal field of view of 67°. They are positioned with a rotation of less than 10°, all pointing to the center of a pentagon with edges 45 cm in length.

3D model reconstruction

Photogrammetry is a process that involves several steps [210]. For each image, the position where it was taken has to be determined. Then the pixels are projected into 3D space. When the equivalent points from different images intersect in space, the 3D position can be

[2]https://www.canfieldsci.com/imaging-systems/vectra-h1-3d-imaging-system/
[3]https://www.canfieldsci.com/imaging-systems/vectra-h2-3d-imaging-system/

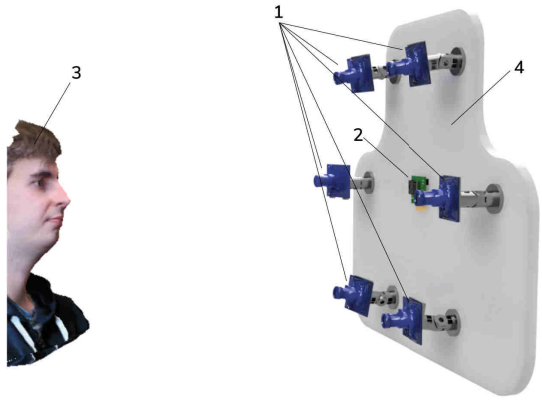


Figure 4.5: Final design of the sensor, (1) cameras, (2) thermal sensor for LFD, (3) person (me), (4) basis

calculated. Other algorithms are used for texture mapping, error suppression, and surface smoothing. The result is a high-quality meshed and textured 3D model.

We have tried open source solutions for this, but the result is that Zephyr's[4] proprietary software produces high quality results/models. The principles of some of the program's key algorithms have been published [211], [212], [213], [214], [215].

With this tool, we have attempted to generate a 3D head model using images captured at different times with a standard photo camera, while the person holds their head in a specific position, see Figure 4.6(a) and 4.6(b). In Figures 4.6(c) and 4.6(d) we can see a 3D mask with texture. The 3D model created from 39 images is already suitable for use, as it includes prominent facial features such as eyes, nose shape, lip corners, and more.

(a) (b) (c) (d)

Figure 4.6: (a) Model created from 11 paired images (input dataset contains 25 images); (b) Model created from 39 paired images (input dataset contains 50 images); (c) and (d) Models with textures with directional light.

We also experimented with images taken at the same moment and we experimented with the six cameras (Figure 4.7). The data from one camera is not used in the final 3D model due to the lack of corresponding points with the other images. The results exceeded our expectations. Only five images captured at the proposed positions are sufficient to create 3D face models. In contrast to the previous scenario where we captured the face from various

[4] https://www.3dflow.net/3df-zephyr-photogrammetry-software/

angles in a sequence, the quality of the output significantly improves when using five images simultaneously taken from precisely defined positions.

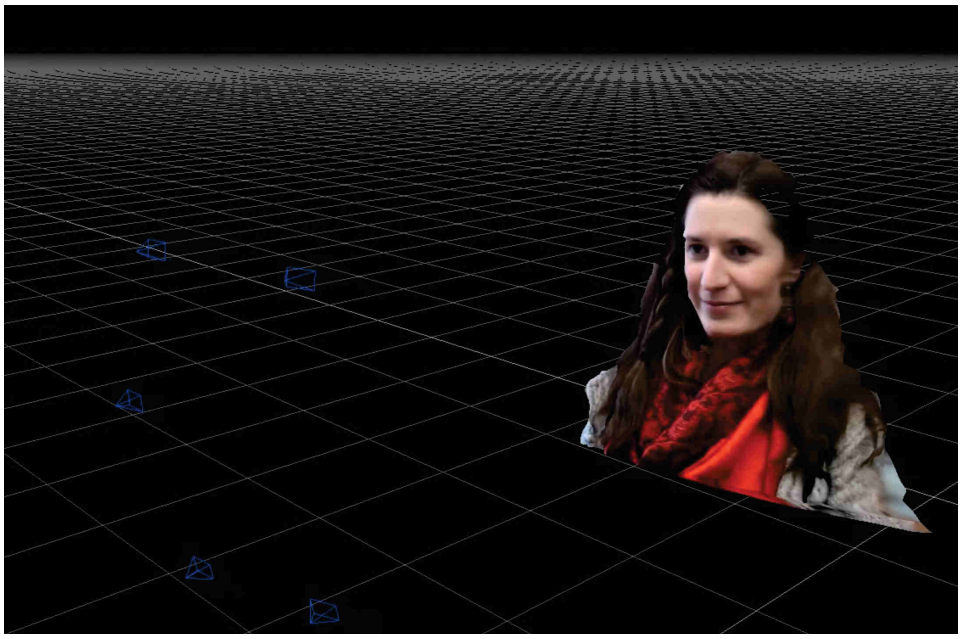


Figure 4.7: First experiment with using six cameras.

Prototype and outputs

The mechanical parts for the scanner were printed on a 3D printer, and the computer for controlling the cameras was placed in the middle, see Figure 4.8. Each of the cameras can be rotated to set or adjust the appropriate angle. The outputs of the scanner is shown in 4.9. The final device is protected by a utility model [TG.5].

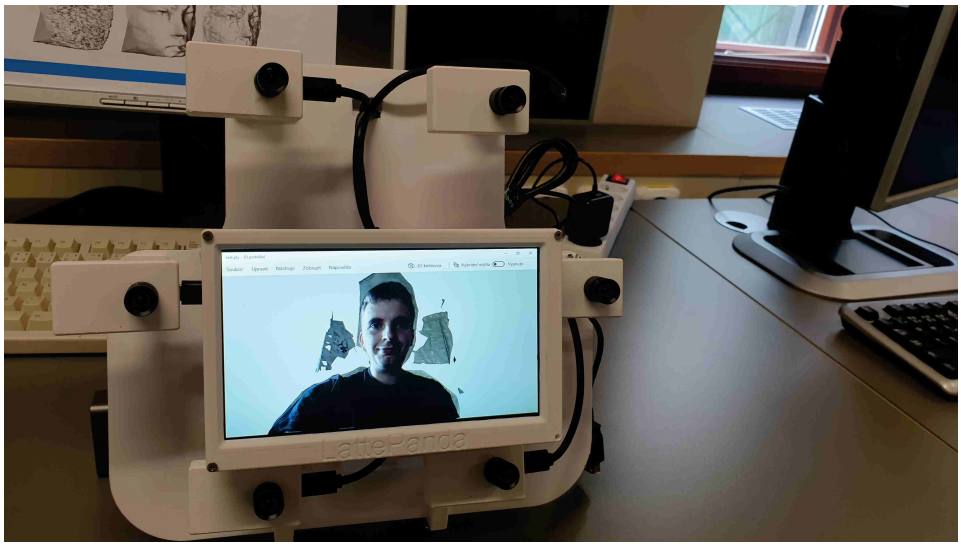


Figure 4.8: Final version of the 3D face scanner.

Figure 4.9: Outputs of our 3D face scanner.

Summary

The primary function of the 3D scanner is to collect a dataset of 3D face profiles. Nevertheless, there are certain drawbacks, most notably the need to use photogrammetry software. Furthermore, the use of the device as a sensor for 3D face recognition is accompanied by extended processing time, in order of minutes even with hardware acceleration. In the future, we expect the development of algorithms that will allow the extraction of 3D-like features from photographs without the need for the reconstruction of a complete 3D model, an advancement that has the potential to open doors for the sensor's application in future 3D face recognition tasks. In the meantime, given the sensor's ability to capture face features, its outputs can still be utilized in face analysis software or, for instance, for the creation of a 2D face image dataset.

4.1.3 Face dataset generator

In the preceding section, we outlined the process of acquiring a 3D head model. Another step in creating a face image dataset generation involves projecting these 3D models onto a 2D plane. Since there was no suitable generator available at the time, we made the decision to create our own solution, which we named SYDAGenerator [TG.6], as introduced in [TG.2]. In developing this tool, we established three primary objectives for the generator to achieve: to simulate the real capture process, to allow the the application of post-processing filters and to enable control of head alignment and rotation.

Moreover, it is worth noting that this generator extends beyond the scope of head models. Although a similar approach has been introduced elsewhere [216], our generator is distinguished by its increased complexity, offering a variety of options for simulating "in the wild" conditions. We have progressively expanded the functionality of this application over time. The set of features include:

- Simulation of camera properties (chip resolution, focal length)

- Possibility to set direction of light and its color
- Rotation of head in three axis (yaw, pitch, roll)
- Setting the background image
- Operations (translation, rotation, scale) to establish the center of rotation
- Width normalization of the head
- Support for post-processing module
- YOLO annotation of Region of Interest (ROI)

The main goal of the generation is to use a 3D model and project it into 2D images with a predefined scene setup. The primary requirement is that the resulting images look as natural as possible. The idea was introduced in [TG.7].

The SYDAGenerator is implemented with the use of the LibGDX[5] library to work with a 3D model and post-processing is solved by algorithms from the OpenCV library. The library supports basic operations with 3D models that are sufficient for the type of application. However, it also brings a limitation, which is a reduced number of vertices to 32K.

The generating process is divided into the following steps:

1. 3D model transformation
2. Scene settings
3. Generate images
4. Image post-processing and filters

Verification of the plausibility of the simulation of the camera properties is attached in Appendix B.1.

3D model transformation

Most scanned 3D models are not suitable for direct use. The SYDAGenerator supports basic transformations to align a 3D model, including translation, rotation, and scaling. If more advanced modifications are required, the model must be processed by an external application (e.g. Blender[6]). During the generation, the 3D model is rotated in yaw, pitch, and roll angles. However, the center of gravity of the model should be selected before starting the generation.

In the case of the head model rotation, determining the center of rotation is a complex problem. The human head rotates through the cervical spine (seventh vertebrae C 1-7) [217], but face or head detectors only capture the head itself. In addition, the curvature of the spine has many degrees of freedom and it would be difficult to determine the rotation derived from the cervical spine. Therefore, the center of gravity of the head must be approximated for simplicity.

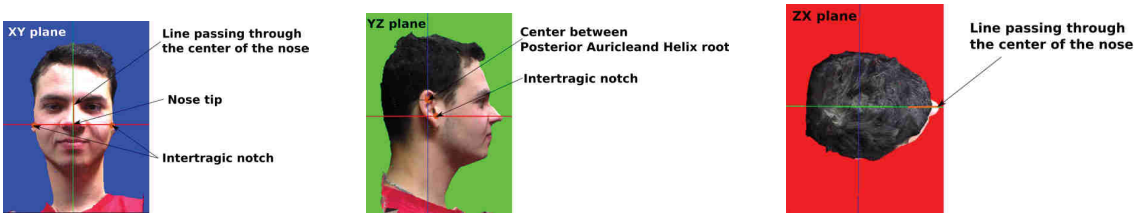
[5] https://libgdx.badlogicgames.com
[6] https://www.blender.org

For correct operation of the generator, a center of gravity, which servers as the center of rotation, must be defined for every model. This center can be determined by the intersection of three planes, as illustrated in Figure 4.10.

In [TG.2], we proposed the following algorithm to define the center of rotation of head.

The first plane is defined as the one passing through the center of the nose. However, defining the second plane is challenging because it is unrealistic to locate nodal points on the face for its determination. We have chosen to use a plane defined by the tip of the nose and the anthropometric point of the ear, known as the *intertragic notch*, which is perpendicular to the first plane. To define the position of this second plane, we need to know the coordinates of these two points and ensure its perpendicularity to the first plane. Alternatively, the second plane can be defined in the same way as the Frankfort horizontal (Section 2.1.2).

The third plane passes through a specific part of the ear, defined as the midpoint between the posterior auricle, helix root, and the tip of the nose. This plane is also perpendicular to the other planes. The center of the head is then established at the intersection of these three planes, as shown in Figure 4.11. This center must be defined for each head model individually.



(a) (b) (c)

Figure 4.10: Planes defining the center of the head rotation.

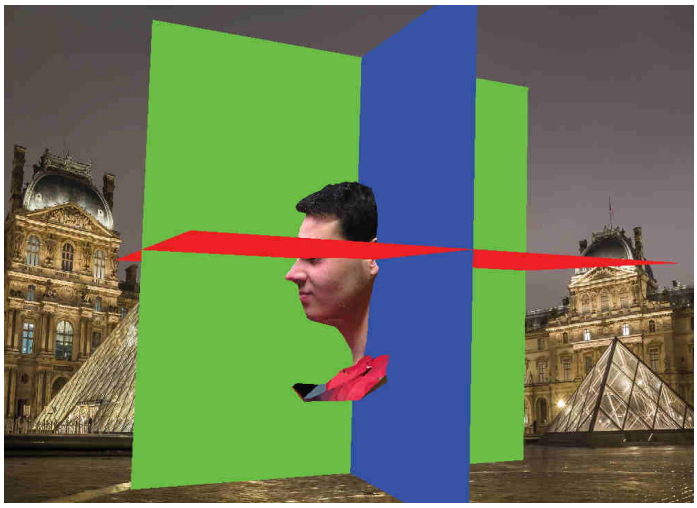


Figure 4.11: The center of rotation of the head model defined by the intersection of planes.

Once the desired transformations are set, it is possible to save the transformation matrix to a JSON file so that the desired matrix is loaded when the program is started. After

running the program, the configuration file with transformations will be loaded if it exists. This is an important feature for automatic generation.

Scene settings

The preceding step involves preparing a 3D model, while the subsequent one revolves around setting up the scene to ensure the dataset conforms to specific requirements. Since the application mimics face capture by a camera, configuring camera properties and lighting becomes imperative. Furthermore, the 3D model is positioned in front of a background derived from the image and is seamlessly projected into the scene without any alterations or distortions.

Another important aspect is the adjustment of the light settings. By default, the tool uses only ambient diffused light to illuminate the scene, where the color temperature can be set. However, if we need directional light, we can set it in the configuration file. However, if the model has deformations on the surface, the directional light will cause highlight artifacts. The software can also adjust the camera position, however after experimenting with the human head, we came to the conclusion that it is better to keep the setting when the camera has a straight direction.

Generating images

The primary function of the application is to generate images with rotated heads, creating a new dataset of face images. To achieve this, the application allows the transformation of a 3D model during the generation process. The 3D model can be rotated in yaw (ψ), roll (ϕ), and pitch (θ) according to the settings (Figure 4.12).

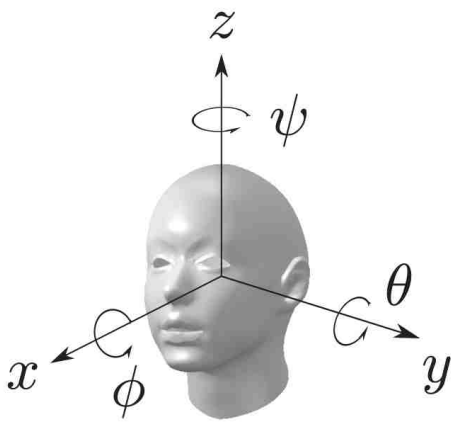


Figure 4.12: Euler angles [218].

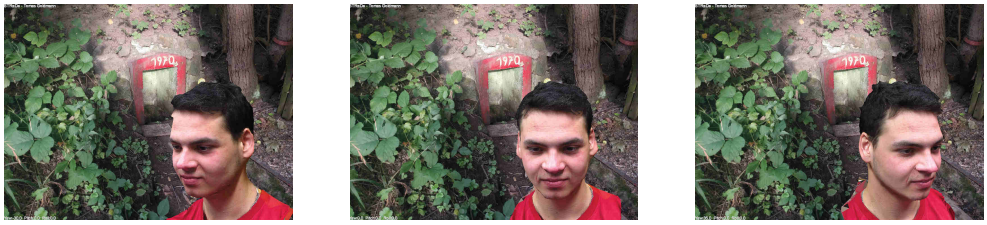
While the 3D head alignment uses quaternions to rotate the object, the image generation is performed using Euler rotation. Euler rotation is defined by the following equations [218]

$$R(\phi, \theta, \psi) = R_z(\psi) R_y(\theta) R_y(\phi) \tag{4.1}$$

where R is the rotation matrix and R_z, R_y, R_x are the partial rotation matrices, defined as follows [218]

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}$$
$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}. \quad (4.2)$$
$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The number of images depends on the step size and the angle range. In addition, the background can be changed for each iteration of the rotation. Examples of images generated by SYDAGenerator are shown in Figure 4.13.



(a) (b) (c)

Figure 4.13: Examples of the generated face image with rotated head.

Image post-processing

A real face image contains unwanted distortions due to camera characteristics and environmental influences. Camera characteristics that affect the image include chip resolution, focal length, lens distortion, and more. If we want to test algorithms for detection and recognition, we should take these paradigms into account. The SYDA generator supports image modification using image processing algorithms. This part is useful for testing of face detectors and classifiers.

For example, the surveillance systems produce images of varying quality, which can negatively affect face detection and identification. We have chosen one property that affects the success of face detection - *resolution*. Using a downscaling algorithm, we can create a dataset that simulates the real conditions of a surveillance system. This dataset proves valuable for assessing the impact of resolution on face algorithms.

Other camera characteristics include *lens distortion* (Equation 4.3), which can have a significant effect on the captured image. Because distortion can alter important facial features, it can degrade or prevent the identification of a person. The distortion filter is determined by mapping pixel values from ideal coordinates to distortion coordinates, defined as follows [219]

$$\begin{bmatrix} X_d \\ Y_d \end{bmatrix} = \begin{bmatrix} X_u \\ Y_u \end{bmatrix} \left[1 + R_u^2 K_{\mu 1} + R_u^4 K_{\mu 2}\right] \quad (4.3)$$

where $R_u = \sqrt{X_d^2 + Y_d^2}$, $K_{\mu 1}$ and $K_{\mu 2}$ are the distortion coefficients, X_d and Y_d are the undistorted image coordinates.

(a) (b) (c)

Figure 4.14: Examples of generated downscaled face images with rotated head.

Summary

The implementation of a tool to generate synthetic or semi-synthetic datasets was driven by the need to extend existing datasets for the evaluation of face recognition algorithms. The SYDAGenerator provides the ability to use 3D head models obtained by modeling, generating, or scanning a head to create a 2D face image dataset. We developed our own sensor capable of acquiring 3D head scans.

In 2018, we started a collaboration with Masaryk University, Faculty of Science, where a dataset consisting of 3D models was created, namely Fidentis [220], as part of their research. The dataset was created by long-term scanning of different individuals under laboratory conditions. This provides an excellent opportunity to use our generator to obtain a unique semi-synthetic dataset for testing face recognition algorithms. While our own 3D scanner is fully operational, recreating a similar dataset would be unprofitable.

Overall, the SYDAGenerator[7] is still available and maintained.

4.2 SYDA-Fidentis dataset

Depending on the type of 3D head model, the SYDAGenerator can be used to generate a synthetic[8] or semi-synthetic dataset[9]. In contrast to the fully synthetic dataset, the main advantage of the semi-synthetic dataset is the preservation of the facial features. This led us to use the SYDAGenerator in combination with the 3D head models of real people from the Fidentis dataset [220] to introduce the new dataset to test the face recognition algorithms. The advantage of the Fidentis dataset is that the scanned heads are aligned based on the Frankfurt horizontal, ensuring that each model has the same center of rotation.

Unlike a synthetic dataset, the images in the proposed dataset should be closer to the real face images. However, despite the best efforts to ensure faithful images, depending on the generator settings, the images may contain deviations from the natural face appearance caused by the generator and model preprocessing.

This generator was designed to simulate the conditions that occur when a real face is photographed. For our dataset, we established the configuration includes properties of the Nikon D2700 camera, such as the dimension of the virtual sensor and the resolution. The focal length was set to 35 mm. From the perspective of the face detector, it was stated in Section 3.4 that the detector may suffer from sensitivity to the size of the face in the image. To better understand the state-of-the-art detectors, we decided to generate the dataset con-

[7] https://www.fit.vutbr.cz/~igoldmann/app/sydagenerator/.en

[8] Dataset is generated using generated 3D head model.

[9] Dataset is generated using a 3D head that is obtained by scanning a real head.

sisting of three subsets with different camera to subject distance (CSD) (4, 7 and 10 meters) between the face and the camera, where each subset includes images rotated in the yaw and pitch axis. Due to the large number of adjustment degrees provided by the SYDAGenerator, the enormous combination of parameters such as light color, light direction, and background can lead to data explosion. Therefore, for this dataset, both distance and rotation in pitch and yaw were selected as variable parameters. Other parameters are frozen and used across all images.

In summary, we created our dataset, named SYDA-Fidentis, using semi-synthetic face images designed to mimic conditions found in real-world environments. To accomplish this, we selected 216 3D head models from the Findetis dataset, consisting of 131 female and 85 male subjects. Figure 4.15 visually represents the age distribution of the subjects in our dataset.

In contrast to the generated 3D models (see experiment in Section 4.1.2), the head models preserve the facial characteristics of real people, each of the heads is aligned according to the Frankfort plane, see Section 2.1.2. The example images are shown in Figure 4.16.

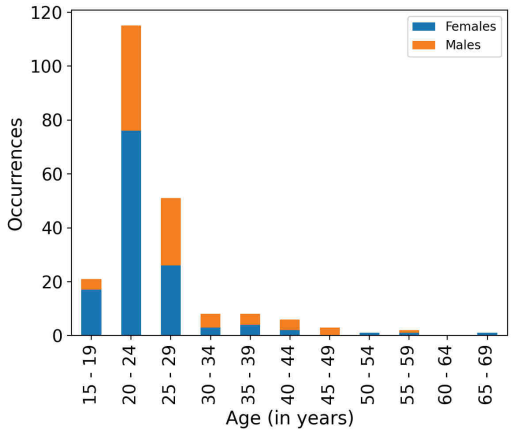


Figure 4.15: Age distribution in the SYDA-Fidentis dataset.

From the perspective of face recognition, the generated dataset needs to be evaluated to discover its properties and to compare the behavior of the face recognition algorithms when interacting with the generated dataset, against their behavior with the real images. As a result, we have established the following research objectives:

- *Determine the impostor and genuine distributions* - the objective is to determine the behavior of the face recognition algorithm applied to the generated dataset. The experiment is based on the assumption that if the face recognition algorithm can discriminate positive and negative matches similarly to real-world datasets, then we can claim that the algorithm is able to identify the necessary features in the images that correspond to face identity.

- *Ability of face detectors to detect faces in the generated dataset* - by comparing the generated images with real face images at different positions, we can estimate the behavior of the generated rotated face images with respect to the real face images. Furthermore, the influence of face rotation on face detection can be evaluated.

- *Face recognition capability within the generated dataset* - similar to the previous goal, but using the face recognition algorithms instead of the face detectors, we can perform

experiments to find out how much face recognition is affected using a generated dataset against a dataset of real images.

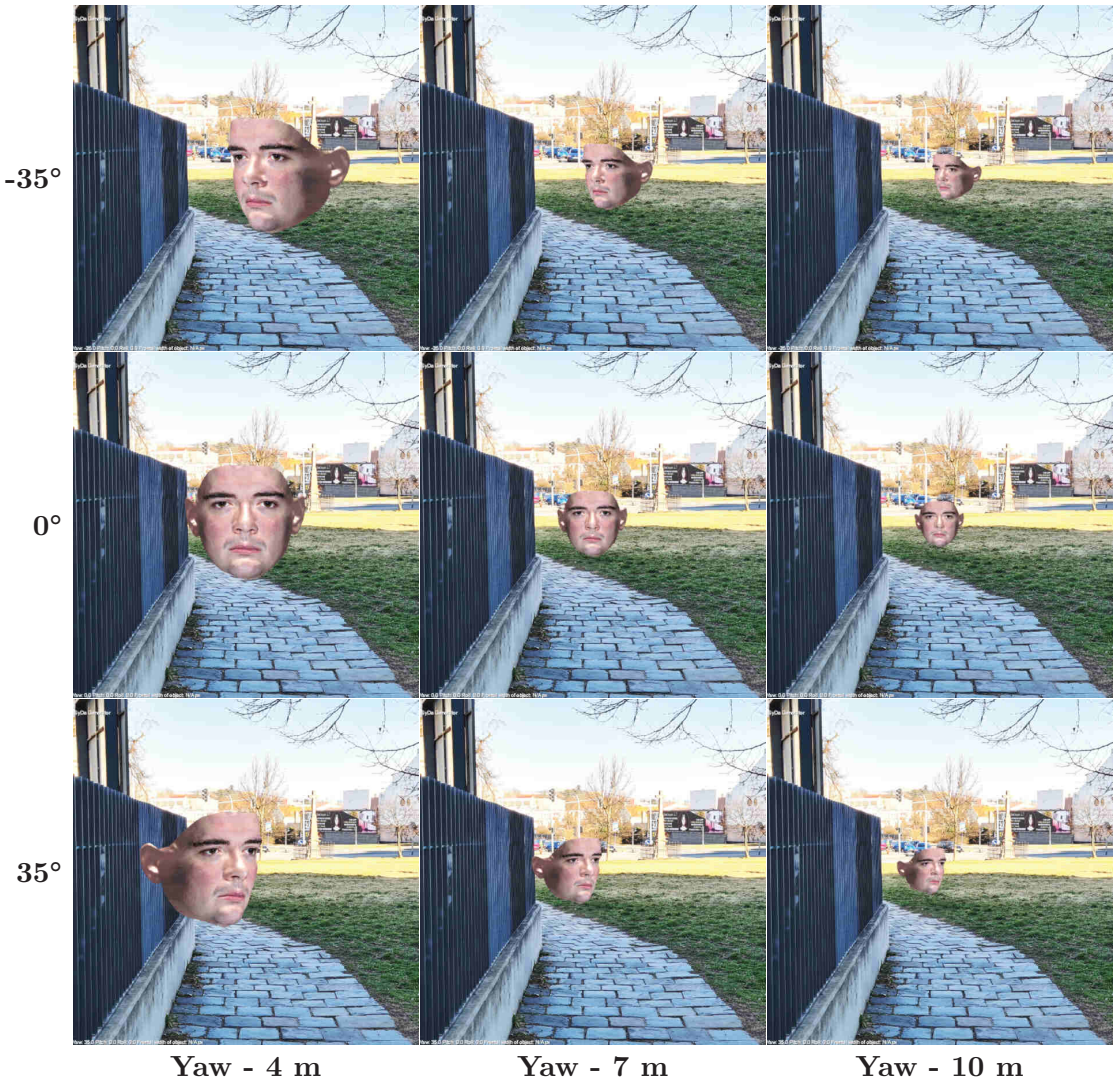


Figure 4.16: Examples of images from our dataset.

4.3 Impostor and genuine distributions of the generated dataset

The first experiment is designed to assess the ability of the generated datasets to discriminate between impostors and genuine matches. As explained in Section 3.5.1, determining the membership of a face image sample to a specific identity relies on a distance metric. When the distance between the input sample and the template is below a certain threshold, we assume that the input sample belongs to the same class. Conversely, if the distance exceeds the threshold, the comparison is classified as negative. However, in the case of the QMagFace algorithm (Section 3.5.10), the genuine and impostor distribution is reversed due to the quality-aware metric.

Furthermore, the distribution for positive matches (impostor) and negative matches (genuine) can be calculated. To balance the two distributions, we used an equal number of positive and negative pairs, following the protocols defined by the datasets authors. For better visualization, the distributions are smoothed using *kernel density estimation* (KDE) [221].

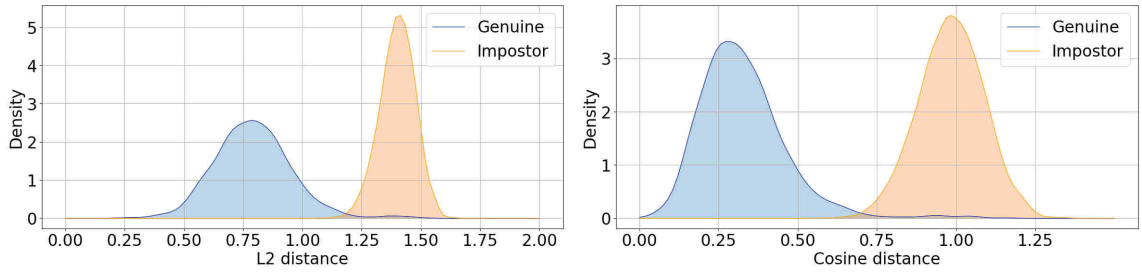
Using the distributions, the ability to discriminate the samples can be observed and evaluated within the dataset consisting of the face images. However, such a comparison is not sufficient to evaluate the similarity between the generated dataset and the real face dataset. Therefore, in addition to the SYDA-Fidentis dataset, we chose two well-known datasets, namely LFW (Section 3.2.3) and CFP (Section 3.2.5), as the foundation for our comparison. For the CFP-FP scenario, a protocol is defined that comprises positive and negative pairs, including frontal and profile images. Thus, we can observe the performance of the face recognition algorithm with images that are captured in profile position compared to frontal position.

ArcFace (Section 3.5.8) and QMagFace (Section 3.5.10) have been chosen as representative face recognition algorithms. For the ArcFace algorithm, the pairs are compared by L2 distance and cosine distance. The experiments were performed first with the LFW and CFP datasets and then with the generated datasets.

4.3.1 LFW and CFP datasets

Let us examine a dataset consisting of real face images. The genuine and impostor pairs were established using LFW testing scenario and the face images were aligned using five facial landmarks.

Initial experiments were carried out using the LFW dataset with ArcFace recognition algorithms, resulting in expected distributions with minimal overlap as shown in Figure 4.17(a,b). Based on this, the appropriate threshold (T) to separate impostor and genuine pairs is set to 1.19, which gives an accuracy of 0.991 (TAR = 0.985, FAR = 0.0034), while for cosine distance distributions, the accuracy is 0.974 (TAR = 0.988, FAR = 0.040) for a threshold set to 0.8.



(a) Pairs matched by L2 distance

(b) Pairs matched by Cosine distance

Figure 4.17: Genuine and impostor score distributions obtained using ArcFace on the LFW dataset.

However, hand in hand with ArcFace, the QMagFace algorithm was used to evaluate the distributions, see Figure 4.18. It is obvious that QMagFace has a better ability to separate the impostors from the genuine matches due to the use of a quality-aware matric.

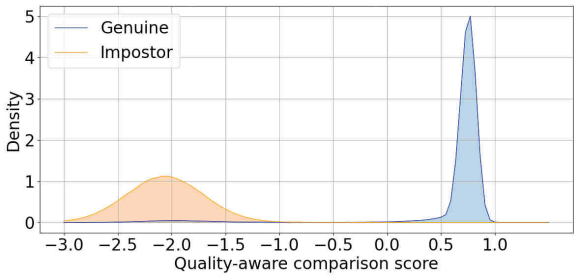
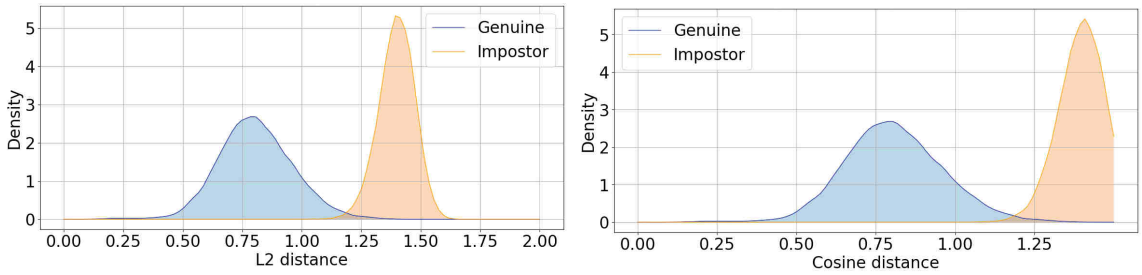


Figure 4.18: Genuine and impostor score distributions obtained using QMagFace on the LFW dataset.

Another representative of the datasets, the CFP dataset, contains both frontal face images and profile face images. The frontal images from the datasets are evaluated according to the same scenarios as the LFW dataset, see Figures 4.19(a,b) and 4.20. In this term, the accuracy obtained by ArcFace (L2 distance) is 0.9797 (T: 1.20, TAR = 0.9732, FAR = 0.0138) and for QMagFace is 0.998 (T: -0.830, TAR = 0.9966, FAR = 0.0002).



(a) Pairs matched by L2 distance

(b) Pairs matched by cosine distance

Figure 4.19: Genuine and impostor score distributions obtained using ArcFace on the CFP frontal dataset.

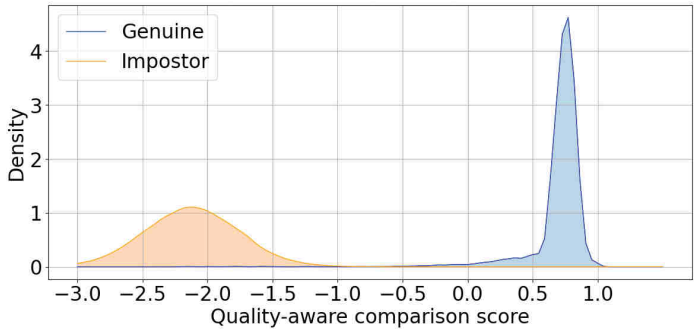
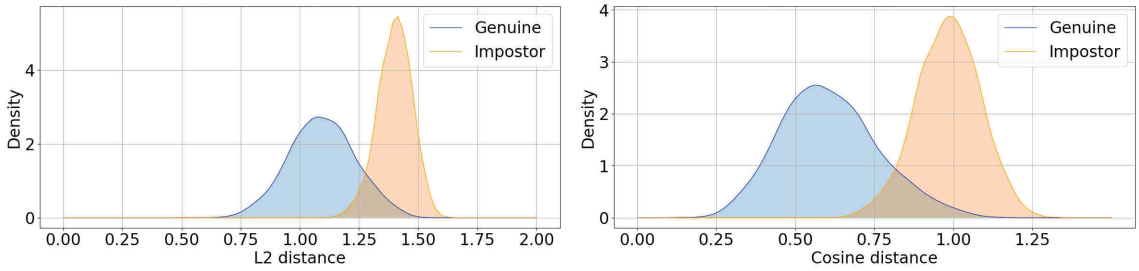


Figure 4.20: Genuine and impostor score distributions obtained using QMagFace on the CFP frontal dataset.

In the last scenario, the CFP-FP protocol is used in order to find the face recognition performance between frontal and profile images. In ArcFace, it is obvious that the overlap of the distributions is larger in the case of the scenario according to the distributions obtained

from the frontal images, Figure 4.21. In this case, the accuracy for the L2 distance matches is 0.97415 (T: 1.28, TAR = 0.9616, FAR = 0.0133).



(a) Pairs matched by L2 distance.

(b) Pairs matched by cosine distances.

Figure 4.21: Genuine and impostor score distribution obtained using ArcFace on the CFP-FP dataset.

Similar to the previous case, the QMagFace algorithm gives better results (ACC = 0.9791, T: -0.99, TAR = 0.962, FAR = 0.00475), and so it is clear that it is better at dealing with faces in non-frontal positions than ArcFace, see Figure 4.22.

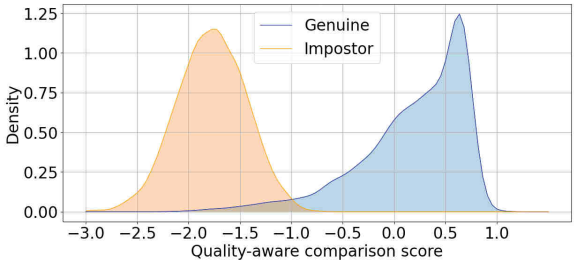


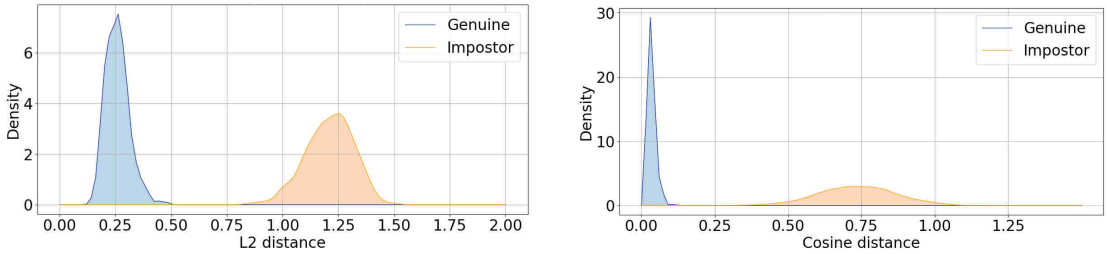
Figure 4.22: Genuine and impostor score distributions obtained using QMagFace on the CFP-FP dataset.

4.3.2 SYDA-Fidentis dataset

The research objective is to find out how the identities contained in the generated dataset are classified based on a metric. Assuming that face recognition has the ability to produce vectors that can discriminate between the impostor and the genuine pairs, we can observe their distributions in order to evaluate the properties of the generated dataset.

Although the LFW dataset consists most of images that capture the face in a frontal position, faces are usually captured at an angle with a small deviation, see Section 2.3.2. Due to the fact that the dataset was generated with heads rotated at different angles, we marked as frontal images with pitch and yaw between 0° and 5°. Similarly, images with a pitch of 0° and 5° were marked as profile images.

Comparisons between frontal images of the dataset showed that although the algorithms were not trained on similar data, both ArcFace (see Figure 4.23) (ACC = 1.0, T: -0.6, TAR = 1.0, FAR = 0.0) and QMagFace (see Figure 4.24) (ACC = 0.9996, T: 0.02, TAR = 1.0, FAR = 0.0008) were very effective at separating the genuine from the impostor distributions. This suggests that there are significant correlations between real and our generated face images.



(a) Pairs matched by L2 distance

(b) Pairs matched by cosine distances

Figure 4.23: Genuine and impostor score distributions obtained using ArcFace on the SYDA-Fidentis dataset (CSD = 4 m).

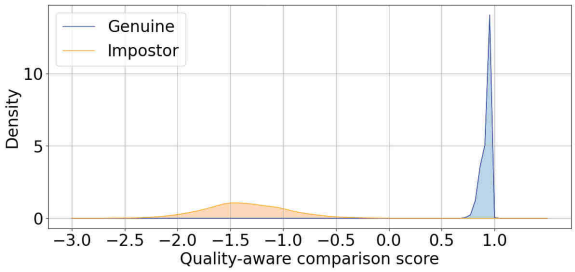
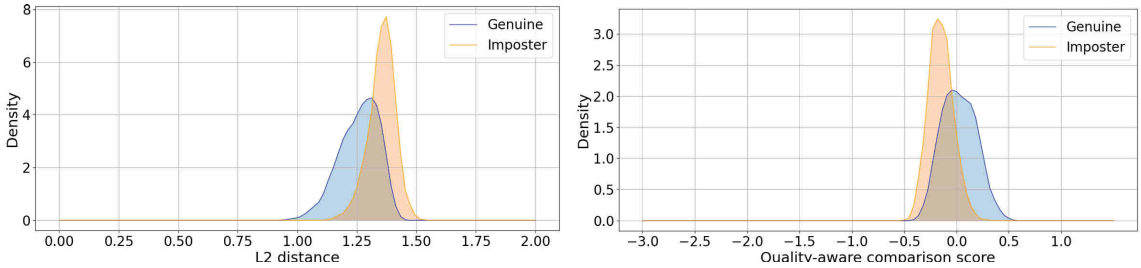


Figure 4.24: Genuine and impostor score distribution obtained using QMagFace on the SYDA-Fidentis dataset (CSD = 4 m).

Next, we conducted experiments to observe how the face recognition algorithms deal with the profile face images. As stated in [137], we consider images in our dataset to be profile images, in which there are faces rotated in the yaw axis between 60° and 90° Each pair consists of a frontal and a corresponding profile image, this is called a frontal-profile scenario. The results are shown in Figure 4.25.



(a) Matches obtained by ArcFace (L2 distance)

(b) Matches obtained by QMagFace

Figure 4.25: Genuine and impostor score distributions obtained using ArcFace and QMagFace on the SYDA-Fidentis dataset, Frontal-Profile (FP) scenario.

The generated near profile face images suffer from poor quality due to corrupted or missing texture. As a result, such images of the SYDA-Fidentis dataset appear to be unsuitable for face detection and recognition algorithms.

Overall, ArcFace and QMagFace behave similarly on generated frontal images as they do on real face images. Although the algorithms have not been trained on such a dataset, the discrimination between genuine and impostor pairs is guaranteed. However, the face

recognition ability of the profile-generated images needs to be further investigated. Therefore, in the following section, we evaluate the ability to detect and recognize faces in various poses.

4.4 Influence of head rotation on face detection

In previous experiments with the SYDA-Fidentis dataset, we have shown that face recognition algorithms are primarily able to discriminate between the generated frontal faces. However, the next research question is how face detection algorithms work with the generated data. First, we investigate the behavior of face detectors on real face images captured at different angles. Overall, the main goal of the following experiments is to observe the influence of head rotation on face detection performance. Within the head rotation, three axes of rotation are used, consisting of yaw, pitch, and roll, see Figure 4.26.

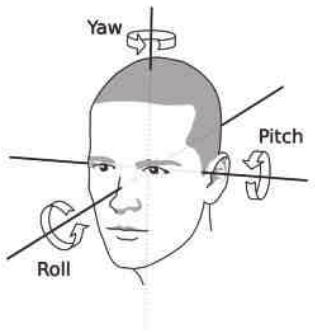


Figure 4.26: Head rotation axes [222].

Due to the ability to correct for roll rotation using the warping transformation [223], the generated dataset consists only of images with faces rotated in yaw and pitch, as shown in Figure 4.27. To remove the roll, it is possible to simply determine the roll rotation according to the eye positions [224], and then modify the image using the affine warping transformation. The roll is calculated by

$$\phi = \arctan\left(\frac{le_y - re_y}{le_x - re_x}\right) \tag{4.4}$$

where le expresses the coordinate of the center of the left pupil (*midpupil*) for both x and y directions and $re_{\{x,y\}}$ is the coordinate of the center of the right pupil (*midpupil*) for both x and y directions.

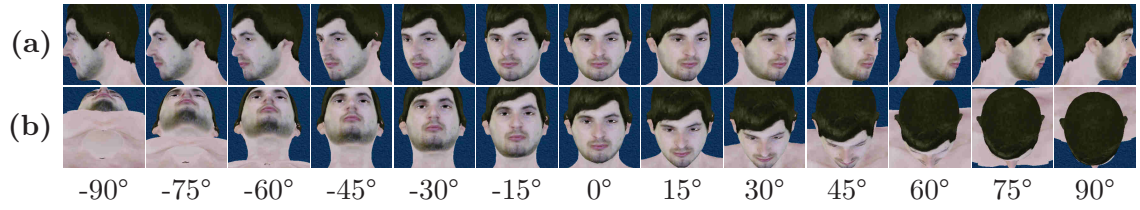


Figure 4.27: Illustration of the rotated head in pitch (a) and yaw (b).

According to the information in Section 3.4, the state-of-the-art face detectors and Viola-Jones from the OpenCV implementation were chosen. Fear not, the Viola-Jones was chosen as a representative of historical approaches, and its contribution to face detection is insignificant at this time.

Experiments were performed with the available implementations of the following detectors: Viola-Jones[10], MTCNN[11], SSD[12], RetinaFace[13], Yolov7-Face (model s), Yolov7-Face[14] (default model) and SCRFD[15] (model 34GF). The MTCNN implementation chosen for the MTCNN detector is a derivative of FaceNet's MTCNN, and the model used is an adaptation of the original implementation. The Single Shot Detector (SSD) is based on the reduced ResNet-10 backbone [225]. The Yolov7 Face is derived from YOLOv7 and has been modified for binary classification and landmark regression. In the case of RetinaFace and SCRFD, corresponding implementations are included in the Insightface Face Library[16].

In terms of state-of-the-art face detection algorithms, we focused on YOLOv7, RetinaFace and SCRFD, which were trained on the WIDER Face dataset. In addition, for the YOLO detectors, the training dataset was extended with the Yolov7-face-label[17] dataset.

Given the available performance results for the selected face detectors (ROC, F1 score), we focused only on evaluating the influence of head rotation on detection performance, which is not evaluated for the selected face detectors. Since there is only one face in the image, the figures only show the recall metrics, for more details on the performance metrics, see Section 2.3.3.

4.4.1 HeadPose

For our purposes, we need a dataset with well-defined face positions. Unfortunately, such datasets are rare or unavailable. We discovered the M2FPA dataset [226], which contains 397,544 images from 229 subjects, with each image featuring a face rotated to a precise position. Access to this dataset requires that you agree to certain terms. We have therefore accepted the agreement and submitted two requests for access to the dataset, but have not yet received a response from the authors.

For our experiments, a dataset called HeadPose (see Section 3.2.5) was chosen as a representative of the real face dataset. The dataset consists of 15 subjects with face images taken at different angles. However, the dataset does not have a well-defined center of rotation, which causes some variations in the results.

To ensure the same conditions for yaw and pitch, we kept the detector's default settings for both rotations.

Face detection in images with yaw head rotation

First, face detection was performed on the images with faces rotated around the yaw axis. Due to asymmetric facial features (such as freckles or pigmentation) [227], the progression of the function may not be the same for positive and negative rotations. The worst result

[10] https://opencv.org
[11] https://github.com/ipazc/mtcnn
[12] https://github.com/serengil/deepface
[13] https://github.com/deepinsight/insightface/tree/master/detection/retinaface
[14] https://github.com/derronqi/yolov7-face
[15] https://github.com/deepinsight/insightface/tree/master/detection/scrfd
[16] https://github.com/deepinsight/insightface
[17] https://github.com/derronqi/yolov7-face

was obtained with Viola-Jones (OpenCV), which is in line with our expectations for this detector. The other detectors had no problems with head rotation along the yaw axis, as shown in Figure 4.28. The algorithms did not produce any false positive detections, resulting in a precision of one for all detectors.

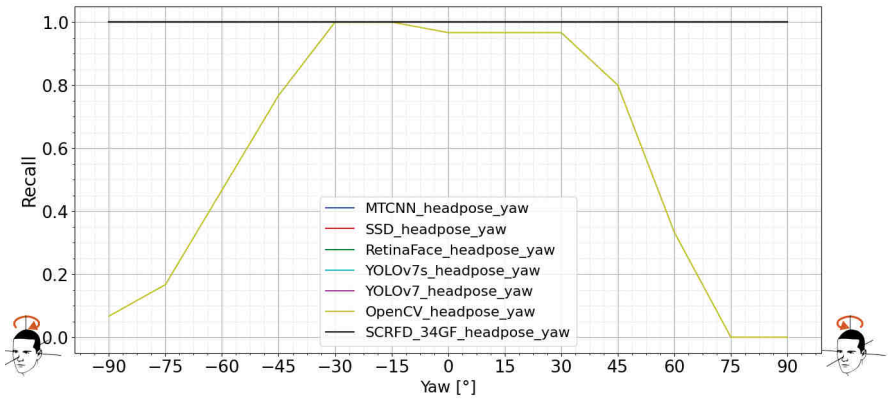


Figure 4.28: Influence of head rotation in the yaw axis on face detection performance in the Headpose dataset.

From frontal to profile, these algorithms expect OpenCV to be able to accurately detect faces, meaning that recall has not decreased.

Face detection in images with pitch head rotation

The pitch rotation is another factor that can distort the features in a face. We found that pitch rotation has a much larger effect on face detection than yaw rotation. Although both YOLOv7 face detectors are trained on the same dataset, their performance differs when the head is rotated in pitch. Although the MTCNN is an older face detector, it outperforms all state-of-the-art detectors in this case.

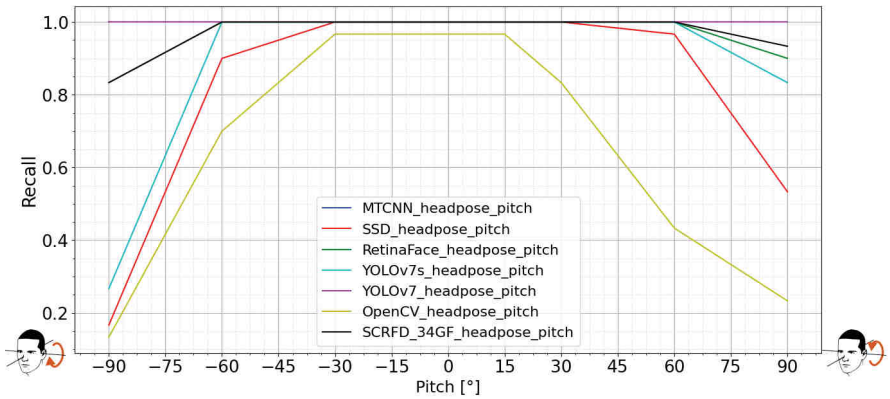


Figure 4.29: Influence of head rotation in pitch axis on face detection performance in the Headpose dataset.

Surveillance cameras should be mounted high and out of reach, such as on building walls [228]. Based on our results in Figure 4.29, detection can be negatively affected by improper angles. More research should be done in this area to improve face detectors.

4.4.2 SYDA-Fidentis

Based on the distribution experiments, we observed that the face recognition algorithms discriminate face images from the SYDA-Fidentis dataset in a manner similar to real face images. The following experiments focus on evaluating the effect of head rotation on face detection. First, the effect of yaw head rotation on recognition performance is evaluated.

Face detection in images with yaw head rotation

In order to investigate the effect of face rotation on face detection performance, the selected face detectors were applied to images from the SYDA-Fidentis dataset, which featured heads rotated in the yaw axis. Experiments were conducted on the SYDA-Fidentis dataset at simulated distances including CSD 4 meters (Figure 4.30), 7 meters (Figure 4.31), and 10 meters (Figure 4.32).

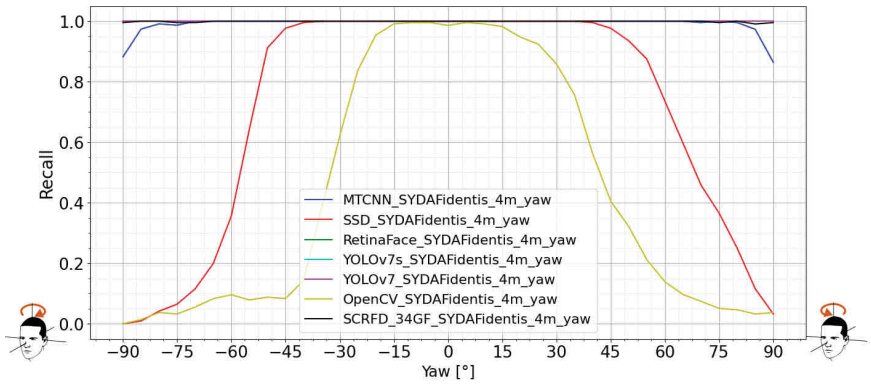


Figure 4.30: Influence of head rotation in the yaw axis on performance of face detection in the SYDA-Fidentis (CSD = 4 m) dataset.

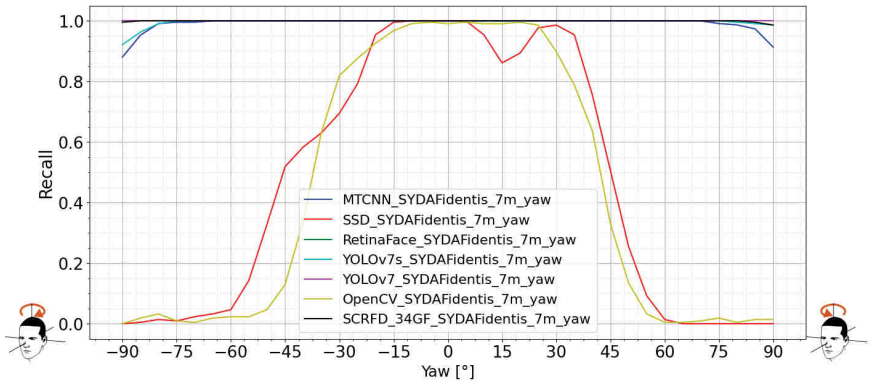


Figure 4.31: Influence of head rotation in the yaw axis on performance of face detection in the SYDA-Fidentis (CSD = 7 m) dataset.

The results show that, as expected, the Viola-Jones detector from OpenCV gives the worst results. The second worst detector among the selected ones is SSD, which is theoretically sensitive to face size. The other detectors can almost perfectly detect the face in the range between -75° and 75°. The observation reveals that even though the generated images only

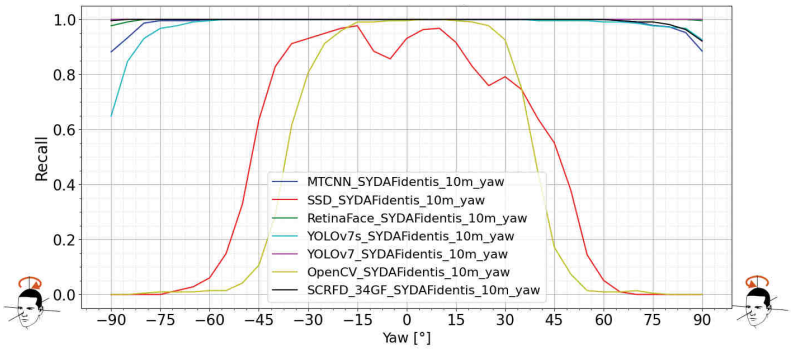


Figure 4.32: Influence of head rotation in the yaw axis on performance of face detection in the SYDA-Fidentis (CSD=10 m) dataset.

contain the texture of the face, without covering the entire profile of the face, the detectors are still able to perform the detection in a reliable manner. Additionally, the detection performance is affected by the size of the face, with smaller profile face having a lower probability of being detected, see Figure 4.32.

Face detection in images with pitch head rotation

Note that the situation becomes more interesting when the faces are rotated around the pitch axis. Obviously, face detection is successful in the range between -45° and 30°, outside of this interval the recall drops rapidly for the most detectors. However, the RetinaFace and SCRFD detectors can handle rotated faces between -60° and 45°, as shown in Figures 4.33, 4.34 and 4.35. These results show the limitations of the generated dataset. The OpenCV and YOLOv7 detectors show an anomaly where the recall rate consistently decreases as the orientation deviates from the frontal position, but begins to increase at a certain angle. This behavior might be a result of the neural networks recognizing patterns that resemble faces located within the face areas.

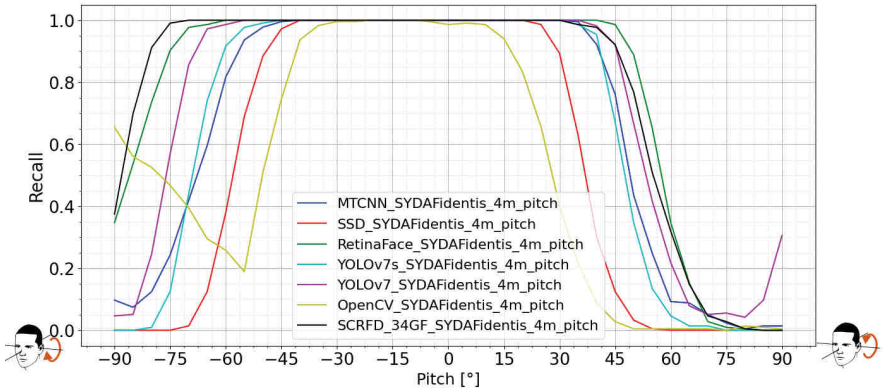


Figure 4.33: Influence of head rotation in the pitch axis on performance of face detection in the SYDA-Fidentis (CSD = 4 m) dataset.

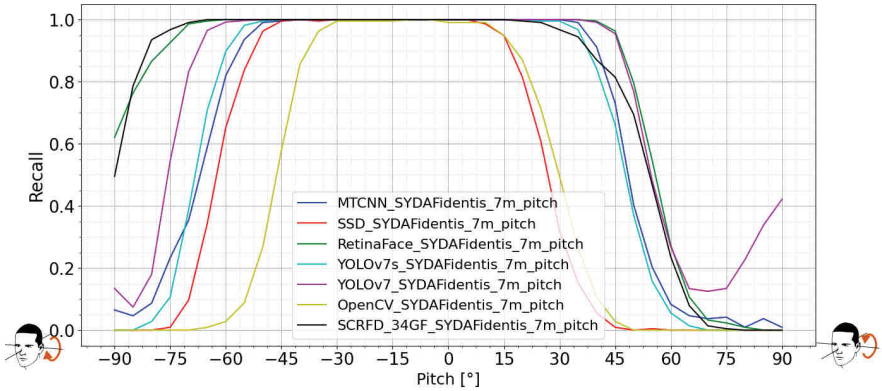


Figure 4.34: Influence of head rotation in the pitch axis on performance of face detection in the SYDA-Fidentis (CSD = 7 m) dataset.

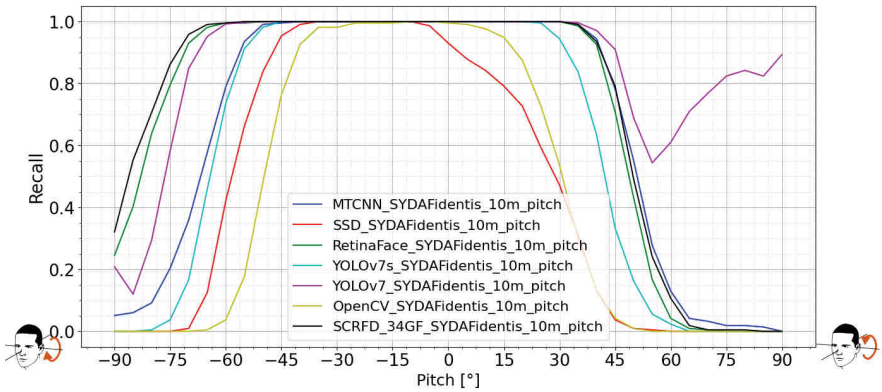


Figure 4.35: Influence of head rotation in the pitch axis on performance of face detection in the SYDA-Fidentis (CSD = 10 m) dataset.

4.4.3 Comparison of face detection between Headpose and SYDA-Fidentis datasets

Based on the results, it is evident that the face detectors exhibit similar responses when applied to both the generated and real datasets. However, there is a slight difference in recall between these two datasets. Except for SSD, the effect of yaw rotation on face detection remains relatively consistent for both the generated and real datasets. When it comes to pitch rotation, the gap between the generated and real datasets widens. Recall performance is similar for both datasets within the range of -45° to 30°, but differs significantly outside of these angle intervals.

The main difference between the datasets is in the positive values of the pitch rotation, where for the dataset with real images, the detection is successful up to 60°, while in the case of the generated dataset this limit is between 30° and 45°.

If the dataset is used within these rotation intervals, it can be employed for testing the detector in various scenarios, such as changes in direction or lighting conditions, lens distortions, and compression artifacts.

4.5 Influence of head rotation on face recognition

The final step in the face recognition pipeline is face embedding, which is responsible for constructing a vector representation that can be used for verification or identification. In recent years, face recognition algorithms have achieved impressive accuracy on datasets with near-frontal face poses. However, we were interested in finding out how accurate these algorithms would be when confronted with images containing other face poses. This ability to evaluate the influence of head rotation on face recognition accuracy is one of the original reasons for introducing our custom face generator. Following the experiments described above, we focus on comparing the accuracy of face recognition by first running the algorithms on the limited dataset of real faces and then on the generated dataset.

For the experiments, we chose well-known algorithms presented in the theoretical part of the thesis, including SphereFace, ArcFace, MagFace, and QMagFace. The RetinaFace detector was used for face detection. We performed both the analysis of the match distances and the evaluation of the accuracy for the selected threshold. With the exception of QMagFace, the threshold was selected based on the error equal rate (EER) of the corresponding ROC obtained from the LFW dataset. In the case of QMagFace, the comparison is conducted using a quality-aware function, the parameters of which are configured based on training samples, FAR, and minimum quality. For QMagFace, the threshold was obtained based on the EER of IJB-C, where its value corresponds to the observations noted in the distribution experiments.

We must not overlook the issue of face alignment, which has a significant impact on face recognition performance. Five landmarks in the face are commonly used for frontal face alignment. These landmarks are aligned as close to the center of the image as possible. However, in the case of the profile images, the transformation causes the nose to be located in the center of the image. As a result, the profiled face image occupies only half of the image. For our purposes, we chose an alignment method that finds similarity between landmarks against reference points.

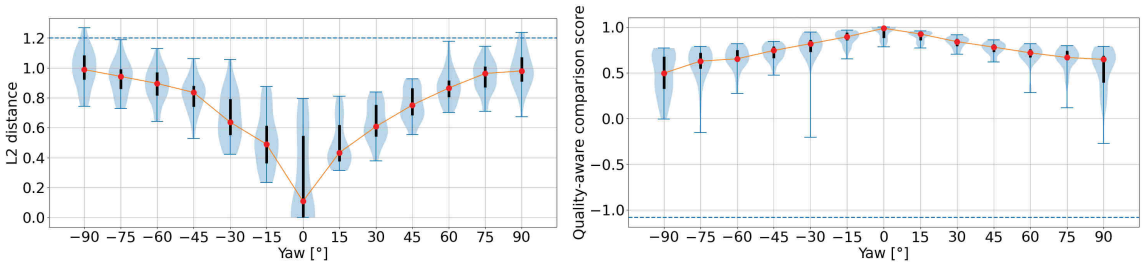
4.5.1 Headpose

First, the capabilities of the recognition algorithms were tested on the face images using images from the Headpose dataset. The goal is to see if and how face rotation affects the L2 distance between the embedding vectors.

Face recognition with yaw head rotation

The first investigation is the recognition performed with the heads rotated in the yaw axis. The position of the face in the images was detected by the RetinaFace algorithm, which gave a high performance in the previous case. Due to the asymmetry of the face, the face rotated in yaw does not match the symmetric values for negative and positive rotation. From the distributions in Figure 4.36 and the accuracy evaluation (Figure 4.37), it is clear that most algorithms, except ArcFace, classified all face images from -90° to 90° below the threshold.

The medians of the distributions (marked in red) gradually approach the threshold as the head rotation moves away from the frontal view. Overall, both detectors are able to handle yaw rotation when using the real face images.



(a) Face recognition performed by ArcFace (b) Face recognition performed by QMagFace

Figure 4.36: Distributions of distances between face vectors with respect to the rotation in the yaw axis.

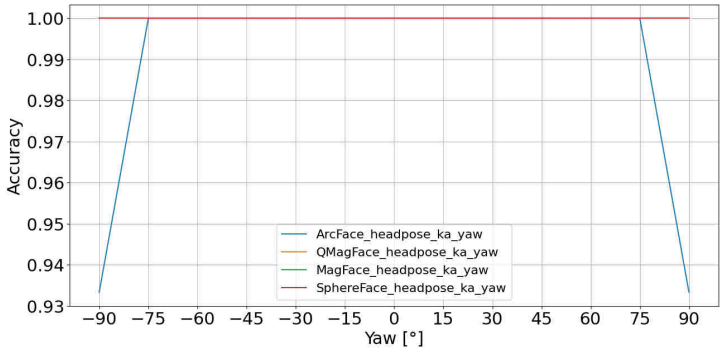
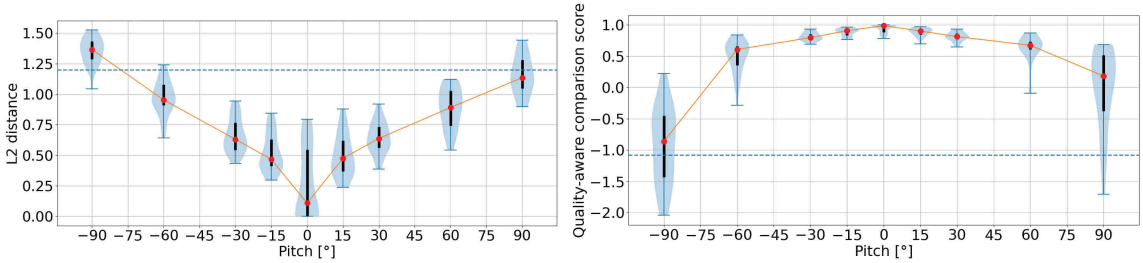


Figure 4.37: Accuracy of face recognition with respect to yaw rotation in the Headpose dataset.

Face recognition with pitch head rotation

The next analysis deals with head rotation in the pitch axis. As mentioned before, since the surveillance cameras are installed at high points, it is important to know the influence of head rotation in the pitch axis on face recognition. Note that the distance distributions are only evaluated for detected faces.

In the case of ArcFace, the values are below the threshold of -60° to 60°, see Figures 4.38 and 4.39.



(a) Face recognition with ArcFace (b) Face recognition with QMagFace

Figure 4.38: Distributions of distances between face vectors with respect to the rotation in the pitch axis.

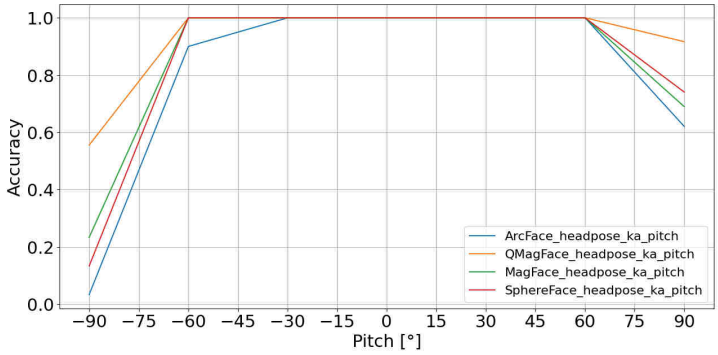


Figure 4.39: Accuracy of face recognition with respect to pitch rotation in the Headpose dataset.

Within the same interval, QMagFace can also correctly recognize a person. However, reliable recognition is only secured in the interval between -30° and 30°. Looking at the head rotation at 60° and -60°, we can see that the median is less favorable for the negative angle than for the positive angle. This means that backward head tilt makes face recognition more prone to accuracy loss. In contrast to the forward head tilt, the face features are visible at higher angles.

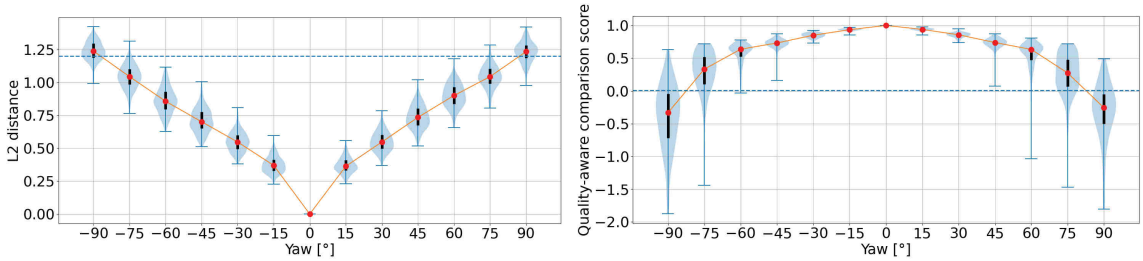
4.5.2 SYDA-Fidentis

From the distribution experiments, it is clear that ArcFace and QMagFace are able to recognize the identities from the generated dataset. However, rotation near the profile position causes a high error rate, so it is worth examining individual distributions for each angle to determine the limits of usable pitch and yaw rotations.

Face recognition with yaw head rotation

The experiments with the head rotated in the yaw axis were performed on the images taken at distances of 4 m, 7 m, and 10 m from the camera. However, the face recognition algorithms scale the ROI with the face to 112×112 pixels, so the results for all distances are very similar. This has led us to present the results for the 4 m distance only.

With ArcFace, all matching pairs are correctly classified in the interval of -60° to 60°, see Figure 4.40. However, for the QMagFace algorithm, a small number of samples exceed the set threshold, resulting in slightly worse performance compared to ArcFace at this angle.



(a) Face recognition with ArcFace

(b) Face recognition with QMagFace

Figure 4.40: Distributions of distances between face vectors obtained from the SYDA-Fidentis dataset with respect to the rotation in the yaw axis.

At -75° and 75° the values are close to the threshold, with some exceeding it for both detectors. This causes a significant drop in accuracy, see Figure 4.41.

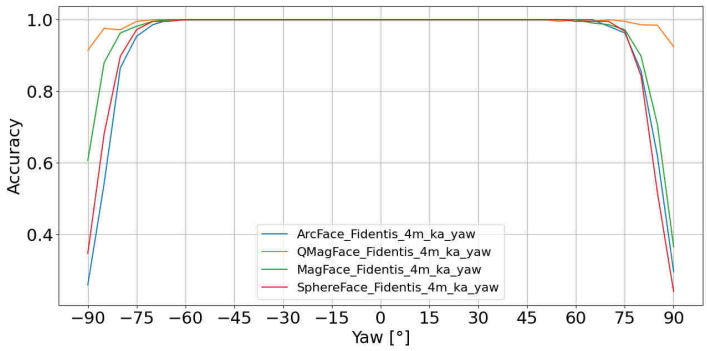
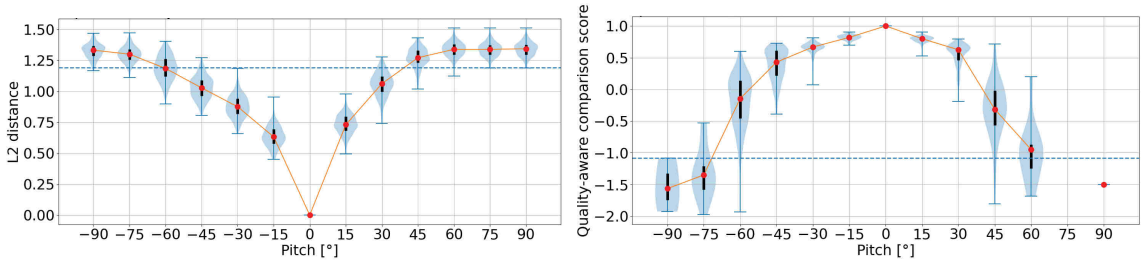


Figure 4.41: Accuracy of face recognition with respect to yaw rotation in the SYDA-Fidentis dataset (CSD = 4 m).

Face recognition with pitch head rotation

The final experiments with the face recognition algorithms deal with a face rotated around the pitch axis. When the face is rotated below -45° along the pitch axis, both algorithms consistently fail to provide accurate identifications. In cases where the angle is positive, the limit falls within the range of 30° to 45°, see Figure 4.42.



(a) Face recognition with ArcFace

(b) Face recognition with QMagFace

Figure 4.42: Distributions of distances between face vectors obtained from the SYDA-Fidentis dataset with respect to the rotation in the pitch axis.

However, an anomaly occurred at 85° when using QMagFace (Figure 4.43), where the distance between the vector generated from the rotated head image and the vector generated from the reference image implies a positive match, while at this angle there should hardly be enough face data present. This observation may indicate security issues, since it implies that it may be possible to generate identity vectors without using real face data.

4.5.3 Comparison of face recognition between Headpose and SYDA-Fidentis datasets

In the case of yaw rotation, the progression of the medians of the distributions in the interval -60° to 60° is similar, with all distributions below the threshold. However, the variance of

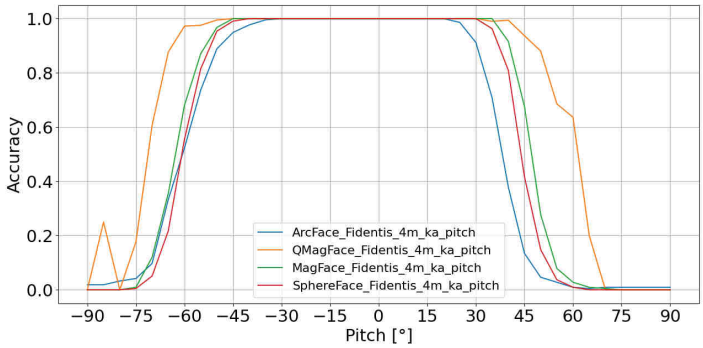


Figure 4.43: Accuracy of face recognition with respect to pitch rotation in the SYDA-Fidentis dataset (CSD = 4 m).

the distributions is larger in the case of the real image dataset, indicating that the features of the faces in the generated dataset are not significantly affected by negative influences.

Rotation in the yaw axis limits the usability of rotated face images outside of the -60° and 60° intervals. Another, more significant limitation is introduced by pitch rotation. The matching distance distributions of faces are similar for real and generated faces in the range of -30° to 30°. The lower limit of usable face rotation is about -45°, while the upper limit is 30°. Beyond these limits, the responses of the tested algorithms on the generated dataset are strongly influenced by incomplete texture or other factors.

Overall, the success rate is directly affected by distorted or missing textures areas when the face is rotated beyond the presented intervals. This led to defining limits for the SYDA-Fidentis dataset.

4.6 Summary

In this chapter, we first focused on the novel possibilities of obtaining a dataset of face images that capture the biological features of real people. As a result, we created a dataset of pitch-rotated and yaw-rotated faces taken at simulated CSDs of 4 m, 7 m, and 10 m using a "virtual" 45 mm lens. For individual CSD, the dataset contains 37 yaw-rotated and 37 pitch-rotated face images for each of the 216 subjects.

The face detection and recognition algorithms were run on our generated dataset and on a dataset containing real images. In these experiments, we investigated the dependence of face rotation on detection and recognition success rates and the behavior of the algorithms applied to the semi-synthetic generated face dataset.

For validation and testing of face recognition algorithms, only a subset of images from the SYDA-Fidentis dataset can be used at certain intervals, determined by comparing the output of the algorithms applied to the generated dataset and the real dataset. In the results, the images can be utilized between -60° and 60° for rotation around the yaw axis. While for pitch rotated face the interval is between -60° and 60°. Based on this, an assumption was made that by using models with better lateral textures, it may be possible to create a generated dataset that has the same behavior as the real one.

By analyzing the distance distributions, we found that yaw rotation less than -75° and greater than 75° is not appropriate for face recognition due to the distances generated close to the threshold. For both detectors, the usable range of rotated images with reduced reliability is in the interval -60° to 60°. However, the effect of pitch rotation on recognition

is even greater, whereas the only face pitch rotation that has no significant effect on face recognition is rotation from -30° to 30°. Due to the loss of face characteristics in positive yaw rotation, the possibilities of 2D face recognition above 30° degrees are very limited. Future research can focus on face recognition using negative rotation around the pitch axis.

4.6.1 Future work

MagFace introduced the idea of ordering the intra-class vector distribution using the quality of the face image. Further improvement of the face recognition loss function can be achieved by using the face rotation in the loss function during training. Similar to the MagFace loss, the intra-class vectors should be organized by rotation. To solve this problem, it is necessary to ensure that the face image dataset with labeled pitch, yaw, and roll rotation has an adequate representation of differently rotated faces.

We believe that this can be the next step in neural networks for face recognition, which can lead to improved face recognition in non-frontal positions.

Chapter 5

Our approaches and acceleration of face recognition algorithms

In the 1990s, computing resources were severely limited, making real-time face recognition virtually impossible. Over time, computing resources have become more powerful and affordable. By 2010, computing resources were sufficient to perform online face detection and then face recognition. For specific tasks in computer vision, technologies such as *Field Programmable Gate Array* (FPGA) [229] or *Digital Signal Processor* (DSP) [230] have been used to accelerate them. As technology has evolved, computers have become smaller and more powerful. This is where *edge computing* [231] comes in.

With the expansion of neural networks, a new platform has emerged to accelerate their inference. Combined with edge computing, it brings the application of face recognition technology to a new level.

Using such a platform, face recognition can be executed on a device with limited computing power. This technology offers new ways to protect the privacy of face recognition system users, as well as to improve the privacy of citizens with the use of government-maintained surveillance systems. For example, when the set of biometric templates is uploaded to the edge device, the device's output can contain only information about the presence of the person of interest to suppress privacy concerns. This system can be used in security cameras and police surveillance tools such as drones, fixed cameras, handheld cameras, and even directly in cameras mounted on police cars. In deploying these systems, we must balance privacy and security concerns, and we must not neglect the risk of misuse of these systems, while adhering to standards, see Section 2.1.3.

A motivation to develop a new solution for face recognition on edge devices arises from the need for a modern, expandable system suitable for use with security cameras or drone cameras. Based on the summary outlined in the theoretical foundation of the thesis and the available data, we chose suitable algorithms for designing these new face recognition systems.

To develop an application for an embedded system, the algorithms should be improved to reduce processing time while maintaining model accuracy as much as possible. Therefore, we proposed and tested improved algorithms for face detection and face recognition on such devices. A similar solution was presented in the paper [232], however, within the scope of our research, we focused on evaluating the performance of our modified algorithms for face recognition on three acceleration platforms, selecting algorithms currently considered state-of-the-art.

For this part, we have set the following research objectives:

- Design and development of face recognition algorithms for use in embedded systems.
- Design and experiments with a lightweight method to improve the quality of a face image for a face recognition neural network.

5.1 Embedded system for face recognition

One of the goals of the thesis is to implement accelerated solutions to perform face recognition. When we first started working on the thesis in 2016, the acceleration of neural networks and machine learning algorithms was performed by GPU [156], FPGA [233] and DSP. Over time, resources dedicated to neural network acceleration began to emerge. Today, neural network acceleration devices can be integrated directly into the CPU [194].

5.1.1 Architecture of the experimental system

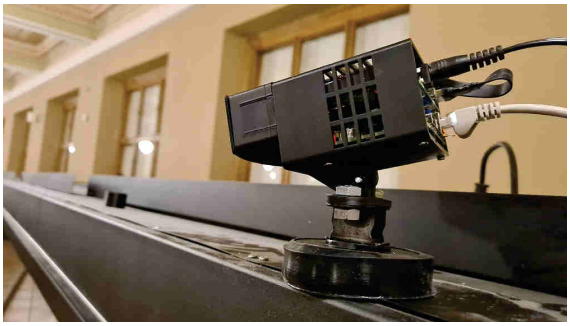
Before delving into the development of the face recognition system itself, we first constructed an embedded camera system, using the Movidius Myriad X platform and UP Core, with power provided by the Intel Atom x5-z8350 CPU, as shown in Figure 5.1(a). Our dedication to advancing this research led us to also explore alternative acceleration platforms. Since the primary focus of our research was low-performance embedding, we switched to an ARM device offering lower performance compared to the Intel Atom. Most notably, we integrated three acceleration devices selected from those described in section 3.6.3, as depicted in Figure 5.1(b).

Today, processors based on the ARM architecture are widely used in Internet of Things edge devices. Therefore, we decided to use the ARM platform as the basis for our solution. To compute the response of the neural network model, we chose three accelerators: Intel Compute Neural Stick 2 (ICNS 2), Hailo, and NPU integrated in RK3568 (2020), which are introduced in Section 3.6.3. The Hailo accelerator is only designed for the M.2 interface or PCIe. Considering these aspects, we chose Rockchip 3A[1] platform, which contains CPU Quad-core Cortex-A55 2 GHz (RK3568) and 0.8 Tera Operations Per Second (TOPS) NPU. To control the accelerators, libraries and Software Development Kit (SDK) are used, which are different for each accelerator.

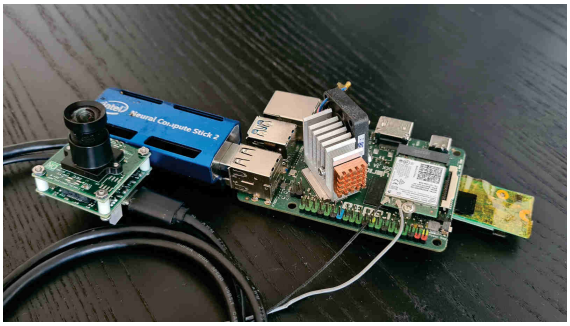
The RetinaFace-based face detector and the ArcFace and MagFace recognition algorithms are implemented in Python using Pytorch and Tensorflow, respectively. The implementations are used to train models for our experimental systems. Furthermore, the trained models are exported to format intended for ICNS 2, Rockchip NPU, and Hailo. The easiest way to infer the model is to use ICNS 2 due to its direct ONNX support. To use the model with Rockchip NPU, the model must be quantized and then saved in RKNN format. Similarly, the Hailo device requires parsing, optimization (quantization), and compilation of the TFLite[2] or ONNX model into the Hailo proprietary format.

The experimental system application is implemented in Python 3.9, which supports all accelerator libraries and SDKs, enabling the implementation of a single application for three accelerators.

[1] https://wiki.radxa.com/Rock3
[2] https://www.tensorflow.org/lite



(a) Our camera with the Myriad X device



(b) Experimental platform

Figure 5.1: Experimental platforms.

5.2 Face detection on embedded devices

The advanced face recognition algorithms discussed in the thesis were developed by modifying the RetinaNet architecture, which can also be adapted for deployment in embedded systems. In addition to their primary task of face localization, these detectors are equipped to perform face landmark regression. Consequently, the detected landmarks can be employed for precise face alignment. To implement these modifications, we used the implementation in PyTorch[3].

The first step is to replace the RetinaFace backbones with more modern architectures that can enhance attributes such as processing time, performance, or the number of model parameters. To achieve this, we utilize various versions of EfficientNet-like architectures. It is worth noting that the use of accelerators may negatively affect the performance of certain operations within the EfficientNet architecture. Due to these potential drawbacks, we expand our selection of EfficientNet backbones to include EfficientNet for embedded systems, known as EfficientNet-Lite.

Except for the Neural Compute Stick 2, the quantized models were calibrated on a subset of WIDER Face images that were first resized to VGA (640×480) and HD (1280×720) resolutions [234][235].

5.2.1 Proposed architectures

The workhorse of the face detector is the three-headed RetinaNet architecture. From the perspective of the RetinaNet/RetinaFace architecture, the feature extraction is provided by two main blocks, the first represented by a backbone that can be easily modified, and the second is the Feature Pyramid Network that performs the upscaling of the feature maps and their merging with the backbone layers. Consequently, the outputs of the FPN are connected to the heads that ensure the detection, for more information, see Section 3.4.7. The layers where the FPNs and heads are connected to backbone, are marked by blue color in Appendix D.1. We have named the RetinaFace detectors based on EfficientNets as *RetinaFace for Edge*, available on Github[4].

Except for ResNet50 and MobileNet0.25 which are original backbones, we focus on the EfficientNet architectures with fewer parameters that seem suitable for use with em-

[3]RetinaFace in PyTorch - https://github.com/biubug6/Pytorch_Retinaface

[4]https://github.com/tgoldmann/RetinaFace_for_Edge - private, available on request

bedded devices, see Section 3.3. An overview of the selected backbones is given in Table 5.1.

Table 5.1: Summary of selected backbones.

Backbone type	#backbone parameters	#RetinaFace parameters
ResNet50	23.508M	27.293M
Mobilnet0.25	0.213M	0.428M
MobilnetV2 1.0	3.504M	6.479M
EfficientNetV2M	52.858M	53.500M
EfficientNetV2L	117.234M	117.869M
EfficientNetB0	5.288M	5.897M
EfficientNetB0Lite	5.288M	5.897M
EfficientNetV2S	20.787M	20.177M

The WIDER Face dataset was chosen for training because it contains a large number of images of varying difficulty, and because it is readily available. In addition, this dataset is commonly used to evaluate the average precision (AP) (Section 2.3.3) of face detectors [154] [173].

5.2.2 Experiments

The face detectors were trained on 100 epochs and assessed using the standard AP metric on the test portion of the WIDER Face dataset. However, due to accelerator limitations, only models that were successfully converted were evaluated.

Performance on GPU and ARM CPU

Performance on the GPU and ARM CPU serves as a baseline for comparing detection on the embedded device. For this reason, in addition to testing the algorithms on the images with the resolution from the test dataset (Table 5.2), the algorithms were also evaluated on images with VGA resolution and HD resolution, see Table 5.3 and Table 5.4. The best results in terms of accuracy are marked in orange. Table 5.2 shows that the original model with Resnet50 is outperformed by EfficientNetV2-S with fewer parameters. However, in the RetinaFace publication, the achieved results on the WIDER Face Hard subset are around 0.91 AP. Thus, the detector with EfficientNet outperforms the published RetinaFace on the WIDER Face Medium subset.

It is interesting to compare the architectures of EfficientNetB0 and MobileNetV2. EfficientNetB0 shows a slight performance advantage over MobileNetV2 in all three subdatasets with images at original resolution. MobileNet 0.25 is typically used in applications where high speed is critical, even at the cost of lower average precision (AP).

For further experiments, we kept the image resolutions fixed at 640×480 (VGA) and 1280×720 (HD), using a letterbox approach mentioned in [236]. In the theoretical part, we noted the abundance of small faces in the WIDER Face dataset, which resulted in a significant gap between face recognition performance at HD and VGA resolutions within the WIDER Face hard subdataset.

Table 5.2: Overview of the AP of the face detectors obtained from the WIDER Face subsets using images in their original resolution.

Backbone	Easy	Medium	Hard	Inference GPU [ms]	MISC[5] [ms]
ResNet50	0.9538	0.9396	0.8430	174	98.0
Mobilenet0.25	0.9070	0.8816	0.7382	38.2	90.4
MobilenetV2	0.9423	0.9248	0.8183	89.2	78.8
EfficientNetV2S	0.9614	0.9614	0.8732	153.4	93.5
EfficientNetV2M	0.9630	0.9501	0.8802	127.3	95.8
EfficientNetV2L	0.9629	0.9536	0.8868	167.7	103.4
EfficientNetB0	0.9444	0.9250	0.8227	91.2	98.5
EfficientNetB0Lite	0.9448	0.9260	0.8353	102.1	90.7

Table 5.3: Overview of the AP of the face detectors obtained from the WIDER Face subsets using VGA resolution images.

Backbone	Easy	Medium	Hard	Inference GPU [ms]	MISC [ms]	Inference ARM [ms]
ResNet50	0.9323	0.8959	0.6391	18.2	35.1	9805
Mobilenet0.25	0.8795	0.8168	0.5280	7.4	27.4	303
MobilenetV2	0.8703	0.7873	0.4421	16.4	31.7	3070
EfficientNetV2S	0.9501	0.9186	0.7087	33.5	36.3	6407
EfficientNetV2M	0.9536	0.9238	0.7174	42.2	34.7	11683
EfficientNetV2L	0.9602	0.9338	0.7368	60.3	36.9	19876
EfficientNetB0	0.9312	0.8894	0.6480	20.0	36.3	1836
EfficientNetB0Lite	0.9350	0.8953	0.6561	30.2	35.2	1854

When examining the results of the WIDER Face Hard, the models using the MobileNetV2 backbone suffered the most from resolution degradation. For the Efficient B0 and Efficient-Lite B0, the decrease in AP was less pronounced.

Table 5.4: Overview of the AP of the face detectors obtained from the WIDER Face subsets using HD resolution images.

Backbone	Easy	Medium	Hard	Inference GPU [ms]	MISC [ms]	Inference ARM [ms]
ResNet50	0.9495	0.9320	0.8217	16.6	82.3	28605
Mobilenet0.25	0.9351	0.9160	0.8021	8.4	85.1	851
MobilenetV2	0.9351	0.9160	0.8021	15.7	70.9	8530
EfficientNetV2S	0.9561	0.9413	0.8560	31.8	91.1	18057
EfficientNetV2M	0.9601	0.9450	0.8628	40.7	80.7	33163
EfficientNetV2L	0.9604	0.9528	0.8723	54.4	79.0	57102
EfficientNetB0	0.9377	0.9208	0.8099	18.7	86.1	5564
EfficientNetB0Lite	0.9391	0.9228	0.8209	27.7	87.0	5499

The EfficientNet backbones seem to be the backbones that provide adequate feature extraction for the RetinaFace detectors. The performance of RetinaFace with the EfficientNet-Lite B0 backbone is slightly worse than that of RetinaFace with ResNet50, while the number of parameters is about five times less.

Performance on the experimental platform

We conducted the following experiments to evaluate the performance level of RetinaFace on embedded devices. It's important to note that this neural network architecture requires an additional post-processing step in order to transform its output into detections, which brings additional computational overhead. This post-processing involves three main operations: first, ordering the detections based on their confidence scores, then filtering through them using a confidence threshold, and finally, applying the NMS (Section 3.4) algorithm to further reduce detections that describe the same object.

In line with our previous experiments, we continued to assess face detection response times and average precision (AP) for both VGA and HD resolutions (as depicted in Figure 5.5). This time, however, we employed the Intel Neural Compute Stick 2 (INCS 2) accelerator, which was connected through a USB interface. In order to perform inference on this device, the models had to be converted from the ONNX format[6] to a format intended for INCS 2, without using quantization.

Table 5.5: AP of face detectors on the WIDER Face dataset and inference time for *Intel Compute Neural Stick 2* accelerator.

Backbone	**Easy**	**Medium**	**Hard**	**Inference time [ms]**	**MISC [ms]**
640×480					
Mobilenet0.25	0.8778	0.8149	0.5268	214.9	91.7
MobilenetV2	0.9211	0.8736	0.6201	382.1	92.2
ResNet50	0.9319	0.8960	0.6392	705.6	83.8
EfficientNetV2S	0.9496	0.9180	0.7081	931.0	81.9
EfficientNetV2M	0.9523	0.9227	0.7168	1545.2	76.7
EfficientNetV2L	0.9598	0.9336	0.7361	2209.2	73.8
EfficientNetB0	0.9311	0.8886	0.6480	540.0	85.6
EfficientNetB0Lite	0.9345	0.8949	0.6560	**531.2**	84.3
1280×720					
Mobilenet0.25	0.9011	0.8719	0.7252	610.2	201.2
MobilenetV2	0.9011	0.8719	0.7252	1289	0.1895
ResNet50	0.9494	0.9318	0.8226	2231	191.3
EfficientNetV2S	0.9556	0.9410	0.8562	2452	180.2
EfficientNetV2M	0.9596	0.9442	0.8618	4094	164.2
EfficientNetV2L	0.9634	0.9509	0.8724	7522	180.1
EfficientNetB0	0.9371	0.9205	0.8098	1472	178.8
EfficientNetB0Lite	0.9384	0.9221	0.8208	1456	151.2

Next, the proposed models were quantized for execution on the Rockchip NPU using asymmetric *uint8* quantization. The models with input images at a resolution of 640×480 px were quantized using the RKNN optimization algorithms. Despite the response time being better than in the previous cases, the AP is affected by the quantization. While all models were successfully quantized for HD resolution, the models based on EfficientNetV2-S, EfficientNetV2-M, and EfficientNetV2-L do not work properly, and their output does not allow face detection.

In terms of acceleration with the Rockchip NPU, the AP is roughly the same for all models. However, the inference time is significantly better than that of the INCS 2 device.

[6] https://onnx.ai/

The best APs are observed with variants of the detector using EfficientNet-V2. Nevertheless, these detectors are slower in terms of inference time. The EfficientNet B0 and Efficient-Lite B0 detectors, which have approximately the same AP as ResNet50, offer better processing times. The results are shown in Table 5.6.

Table 5.6: AP of the face detectors on the WIDER Face dataset and inference time performed on *Rockchip NPU*.

Backbone	**Easy**	**Medium**	**Hard**	**Inference time [ms]**	**MISC [ms]**
640×480					
Mobilenet0.25	0.8560	0.7871	0.5010	108.6	80.8
MobilenetV2	0.9207	0.8723	0.6176	161.8	57.7
ResNet50	0.9311	0.8943	0.6363	288.7	54.4
EfficientNetV2S	0.9481	0.9170	0.7068	284.2	49.7
EfficientNetV2M	0.9473	0.9167	0.7127	430.4	41.8
EfficientNetV2L	0.9563	0.9305	0.7343	623.5	51.2
EfficientNetB0	0.9153	0.8681	0.6066	**187.3**	52.8
EfficientNetB0Lite	0.9270	0.8856	0.6432	**188.4**	57.5
1280×720					
Mobilenet0.25	0.8877	0.8565	0.7036	271.3	163.5
MobilenetV2	0.9048	0.8817	0.7298	462.1	150.2
EfficientNetB0	0.8920	0.8690	0.7020	576.6	168.8
EfficientNetB0Lite	0.8966	0.8760	0.7390	569.3	155.2

The following experiments were conducted with the Hailo device. Before using the device, the model must be converted and compiled. With Hailo SDK, the model can be quantized to 8-bit or 4-bit. In order to maintain the accuracy, the 8-bit quantization was selected. Not all of the proposed models were compiled due to errors in the conversion process. Therefore, Table 5.7 only gives an overview of the compiled models.

Table 5.7: AP of the face detectors on the WIDER Face dataset and inference time performed on *Hailo*.

Backbone	**Easy**	**Medium**	**Hard**	**Inference time [ms]**	**MISC [ms]**
640×480					
Mobilenet0.25	0.8769	0.8139	0.5222	19.3	67.4
MobilenetV2	0.9197	0.8710	0.6180	34.6	64.3
ResNet50	0.9320	0.8960	0.6393	**55.5**	53.6
EfficientNetV2M	0.9460	0.9168	0.7140	236.4	56.3
EfficientNetB0	0.8320	0.7670	0.4590	**71.9**	54.4
EfficientNetB0Lite	0.8924	0.8925	0.5730	**69.5**	58.8
1280×720					
Mobilenet0.25	0.8877	0.8565	0.7035	28.0	148.7
MobilenetV2	0.9339	0.9144	0.8001	85.7	147.4

In general, the Hailo platform has excellent support for ResNet and MobileNetV2 in contrast to the EfficientNet backbones. Using ResNet50 as the backbone, face detection can be performed in approximately 100 ms. Considering all the above possibilities for face

detection, the Hailo with ResNet50 backbone offers the best trade-off between speed and average accuracy.

5.3 Face recognition on embedded devices

From the perspective of face recognition, the angle distribution of features is used by state-of-the-art algorithms. Although first such algorithms were introduced in 2017 [190], this principle is still used in the new algorithms. This led us to use the ArcFace and MagFace algorithms as the engine in our embedded system to perform face recognition.

5.3.1 Proposed architecture

In their original versions, the ArcFace algorithm used ResNet as its feature extractor, while MagFace, which primarily differs from ArcFace in its loss function, utilized IResNet. To optimize these algorithms for embedded devices, we replaced their feature extractors with more recent EfficientNet backbones. Our primary source of inspiration for this implementation was the existing ArcFace codebase. We have also implemented support for MagFace in Tensorflow 2.0. Our implementation is available on Github[7].

All input face images had a resolution of 112×112 pixels and were in RGB format. Further details can be found in Table 5.8, which provides an overview of the selected model backbones along with their respective parameter counts. All models were trained on the MS1M dataset [126] using the SGD optimizer (Section 3.1), with the learning rate tuned according to the model architecture.

Table 5.8: Overview of proposed models.

Abbreviation	Architecture	Backbones	Step decay	#parameters
res50	ArcFace	Resnet50	No	40.376M
efb0flr	ArcFace	EfficientNet B0	Yes	16.412M
efb0flr_mag	ArcFace	EfficientNet B0	Yes	16.412M
efb1flr	ArcFace	EfficientNet B1	Yes	17.068M
efb1flr_mag	ArcFace	EfficientNet B1	Yes	17.068M
efb0lite	ArcFace	EfficientNet B0 lite	No	13.906M
efb1lite	ArcFace	EfficientNet B1 lite	No	14.683M
efb0lite_mag	MagFace	EfficientNet B0 lite	No	13.906M
efb1lite_mag	MagFace	EfficientNet B1 lite	No	14.683M
efV2b0_flr	ArcFace	EfficientNet V2 B0	Yes	16.413M
efV2S_flr	ArcFace	EfficientNet V2S	Yes	30.825M
efV2M_flr	ArcFace	EfficientNet V2M	Yes	63.644M
efV2S_flr_mag	MagFace	EfficientNet V2S	Yes	30.825M
efV2M_flr_mag	MagFace	EfficientNet V2M	Yes	63.644M

The performance was evaluated on the well-known test datasets such as LFW (Section 3.2.3), CFP-FP (Section 3.2.5), and AgeDB30 (Section 3.2.3).

5.3.2 Experiments

The models were trained using the default parameters for ArcFace and MagFace according to their papers. Since the backbones used directly affect the training of the neural network,

[7] https://github.com/tgoldmann/Arcface_Magface_for_Edge - private, available on request

the number of epochs for model training was continuously adjusted according to the evaluation on the LFW, CFP-FP, and AgeDB datasets. The accuracy on LFW was used as a metric for selecting the dataset.

The models were trained using the aligned MS1M dataset (Section 3.2.3). During the initial training of the models with EfficientNet, the training became unstable (see Appendix E.1). Therfore, a step decay with a reduced initial step was used to suppress it. Unlike older backbones, the EfficientNet-like backbones are designed for fast training [151].

Performance on GPU and ARM CPU

The trained models were first evaluated on the NVIDIA RTX A5000 GPU to obtain the reference results for comparison with the quantized models. Likewise, the inference time was measured on the ARM CPU (Cortex-A55, 2 GHz). The results are shown in the table 5.9. The accuracy evaluation was conducted on the test dataset for each epoch and the results of the evaluation can be found in the appendix E.1.

In terms of inference time, the duration on the GPU was consistently below 4 milliseconds in all scenarios. However, we hold the perspective that evaluating the performance on a high performance machine is less important compared to assessing it on a low performance machine, where the differences among the times are significant. Furthermore, the inference times of the ARM CPU used by our experimental platform have been included in Table 5.9, in addition to the accuracy obtained from the test data.

Table 5.9: Accuracy of the ArcFace and MagFace models with EfficientNet and ResNet50 backbones.

Name	LFW (ACC)	AgeDB30 (ACC)	CFP-FP (ACC)	Epoch	ARM inference time [ms]
resnet50	0.994667	0.947167	0.945429	11	391.9
efb0flr	0.991667	0.918000	0.932000	15	123.4
efb1flr	0.994333	0.930333	0.934714	20	189.8
efb0lite	0.993667	0.927500	0.927286	21	97.5
efb1lite	0.994000	0.931333	0.932571	22	126.4
efb0flr_mag	0.991000	0.918667	0.932429	18	125.4
efb1flr_mag	0.993667	0.929333	0.940143	15	176.1
efb0lite_mag	0.993000	0.919667	0.924000	25	97.0
efb1lite_mag	0.994667	0.926333	0.929286	17	123.4
efV2S_flr	0.996167	0.955833	0.957571	10	396.6
efV2M_flr	0.930500	0.714167	0.807429	13	-
efV2S_flr_mag	0.996333	0.960167	0.960571	12	-
efV2M_flr_mag	0.995500	0.960000	0.956000	13	-

The results show that the accuracy on LFW exceeds 99 % for most models. In the case of ArcFace, using EfficientNetV2-S as the backbone, it outperforms ResNet. Similarly, for MagFace, when EfficientNetV2-S or EfficientNetV2-M is used as the backbone, the performance exceeds that of the original implementation, which includes ResNet50 backbones. Furthermore, when we focus on the CFP-FP dataset, we see that the MagFace models with EfficientNetV2-S and EfficientNetV2-M have significantly improved capabilities in detecting images outside the frontal angle. The most noticeable differences among the models are observed in the evaluation on the CFP-FP dataset, and the evaluation of performance is

illustrated by the ROC curves in Figure 5.2. The ROC curves for the LFW and AgeDB30 datasets can be found in the Appendix E.2.

If these neural network models were to be used directly on an ARM processor, approximately 97 ms per face frame would be required for the EfficientNet B0 and EfficientNet B0 Lite models. For ResNet50 and EfficientNet models with more parameters than EfficientNet-V2 S, the processing time would exceed 390 ms.

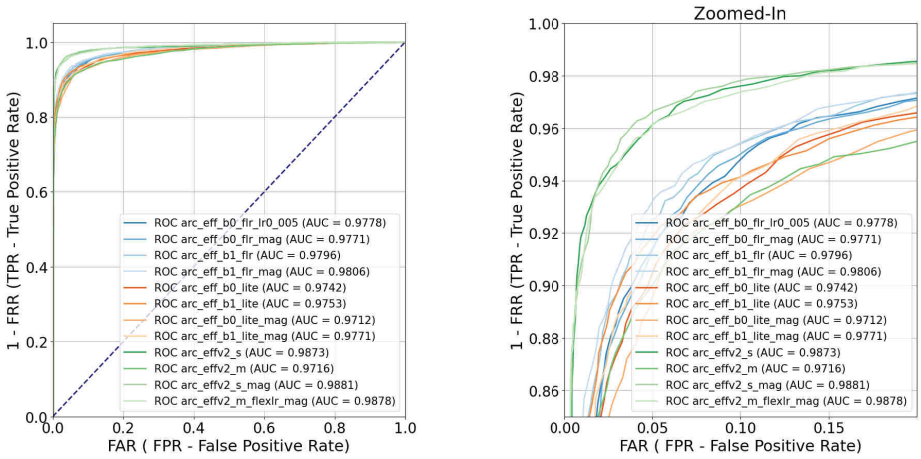


Figure 5.2: Evaluation of the trained models on the CFP-FP dataset - ROC curves.

Performance on the experimental platform

From the experiments with face detectors, it is clear that the performance of quantized models with EfficientNet backbones can be affected by the quantization process. The experiments with the proposed models were first performed using the INCS 2 inference engine, as shown in Figure 5.10.

Table 5.10: Accuracy of face recognition performed with the *Intel Neural Compute Stick 2* device.

Name	LFW (ACC)	AgeDB30 (ACC)	CFP-FP (ACC)	Inference time [ms]
resnet50	0.9948	0,9432	0.9451	44.9
efb0flr	0.9830	0.9010	0.9037	66.4
efb1flr	0.9557	0.8270	0.8940	84.2
efb0lite	0.9930	0.9318	0.9260	42.2
efb1lite	0.9937	0.9288	0.9316	51.6
efb0flr_mag	0.9712	0.8727	0.8847	66.3
efb1flr_mag	0.9782	0.8897	0.8940	84.2
efb0lite_mag	0.9922	0.9177	0.9199	42.0
efb1lite_mag	0.9937	0.9228	0.9289	51.8
efV2S_flr	0.9915	0.9355	0.9424	114.1
efV2S_flr_mag	0.9782	0.8897	0.8940	83.0
efV2M_flr_mag	0.9915	0.9448	0.9391	188.1

The Rockchip NPU outperformed the INCS 2 in terms of processing speed. However, some models showed a significant drop in accuracy, as shown in Table 5.11. Although

the implementation with ResNet50 gives the highest accuracy, the solution with the MagFace model based on EfficientNet-Lite B0, which is able to process a single face image with a resolution of 112x112 in 15.9 ms, seems to be well suited for embedded devices. Apart from ResNet50, this model demonstrates the best performance, even for rotated faces (see CFP-FP in Table 5.11). As a result, the RockChip NPU can process approximately 60 detected face images per second.

Table 5.11: Accuracy of face recognition performed with the *Rockchip NPU*.

Name	LFW (ACC)	AgeDB30 (ACC)	CFP-FP (ACC)	Inference time [ms]
resnet50	0.9948	0.9467	0.9430	26.7
efb0flr	0.5750	0.5073	0.5407	22.2
efb1flr	0.5598	0.4993	0.5607	24.9
efb0lite	0.9928	0.9250	0.9247	**13.9**
efb1lite	0.9927	0.9303	0.9273	**16.3**
efb0flr_mag	0.5383	0.5462	0.5041	23.0
efb1flr_mag	0.5843	0.5328	0.5574	24.7
efb0lite_mag	0.9932	0.9193	0.9244	**14.1**
efb1lite_mag	0.9945	0.9242	0.9300	**15.9**
efV2S_flr	0.7728	0.6543	0.7089	30.0
efV2S_flr_mag	0.8001	0.6572	0.7391	29.9
efV2M_flr_mag	0.6735	0.8285	0.7631	41.7

All models were also quantized for the Hailo device, which is theoretically designed to achieve fast processing with high accuracy. The results are shown in Table 5.12.

Table 5.12: Accuracy of face recognition performed with *Hailo* device.

Name	LFW (ACC)	AgeDB30 (ACC)	CFP-FP	Inference time [ms]
resnet50	0.9852	0.8672	0.9066	20.1
efb0flr	0.6317	0.5005	0.5520	14.9
efb1flr	0.5343	0.4897	0.5119	17.9
efb0lite	0.9482	0.7393	0.8399	6.2
efb1lite	0.9455	0.7432	0.8174	7.3
efb0flr_mag	0.5230	0.5053	0.5146	14.8
efb1flr_mag	0.5197	0.4997	0.5351	18.3
efb0lite_mag	0.9432	0.7398	0.8093	6.6
efb1lite_mag	0.9702	0.7742	0.8761	7.7
efV2S_flr	0.8435	0.6718	0.7233	26.4
efV2S_flr_mag	0.8122	0.6587	0.7159	26.0
efV2M_flr_mag	0.8402	0.6768	0.7670	49.3

In contrast to RetinaFace Edge face detectors, the quantization process is responsible for the significantly lower accuracy of models for face recognition. Considering the Intel Neural Compute Stick 2, the ResNet model stands out as a suitable choice due to its favorable trade-off between accuracy and inference time (AgeDB30 0.9467, 44.9 ms). However, in the case of ArcFace with ResNet using the Rockchip NPU, the performance is similar with an inference time of 26.7 ms. The least accurate model is MagFace with EfficientNet-Lite B1, with an inference time of 15.9 ms.

The absolute fastest processing is achieved with the Hailo accelerator, which can generate the embedding vector in just 6.6 ms for the EfficientNet-Lite B1 model. However, the accuracy is lower than in the previous case. It is important to note that quantization alters the ROC of the model; i.e., the threshold is different from the original model. Consequently, when deploying the model, the ratio between FRR and FAR must be established based on the evaluation accuracy directly on the quantized model for the used accelerator.

Furthermore, we found that using EfficientNetV2-S models can improve face recognition performance in the case of ArcFace and QMagFace algorithms. However, these models should not be used for Rockchip NPU and Hailo due to low accuracy.

5.4 Face image preprocessing

The image quality of face images can be affected by the presence of shadows, blur, and compression. To mitigate these issues, we have implemented a filter based on commonly used computer vision methods to adjust the input images before feeding them into a neural network for processing.

During the research, it was discovered that modern face recognition algorithms only apply normalization techniques to face images for the preprocessing phase, thereby preserving the original image quality along with potentially damaged or low-quality areas.

To enhance the quality of a face image, various types of generative neural networks can be used for preprocessing. We have previously conducted a series of experiments aimed at reconstructing face images using these networks, and the results indicate that reconstruction has a positive impact on the accuracy of face recognition, as we have detailed in our paper [TG.8]. Nevertheless, employing a neural network for image preprocessing demands substantial computing power, making it unsuitable for embedded systems. In addressing the issue of distortion in face images caused by shadows and reflections, we carried out experiments that employed a straightforward local image normalization algorithm. Our primary research focus centered on assessing the influence of kernel size on recognition performance, representing the key contribution of our study.

5.4.1 Filter for suppressing shadows in a face image

One approach for improving non-uniform brightness in images involves utilizing Gaussian functions to calculate the variance and mean of the image, as detailed in [237]. It is worth noting that this method is primarily tailored for grayscale images. Building upon this algorithm, the Gaussian-HSV filter was specifically developed to enhance the accurate recognition of color face images.

In modern face recognition, it is essential to work with color face images. Consequently, the initial step in these algorithms involves extracting color intensity information. Unlike the RGB color model, the HSV model directly expresses intensity through the V channel, which can be can be susceptible to variations caused by uneven lighting conditions. As a result, our approach primarily focuses on adjusting the intensity within this channel. To achieve this, we begin by computing a Gaussian-blurred image through convolution of the input channel I with the Gaussian filter G, defined as follows

$$I_G = I^2 * G = \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} I[x,y]^2 * G = \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \sum_{u=-k}^{u=k} \sum_{v=-k}^{u=k} I[x,y]^2 G[x+u, x+v] \quad (5.1)$$

where W and H are the image width and height, respectively. The Gaussian filter G is given by [238]

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{5.2}$$

Our filter for improving the image brightness is mathematically defined as follows

$$I_{enhanced} = \frac{I}{\sqrt{I_G} \cdot \alpha} \tag{5.3}$$

where the parameter α controls the variance.

While it is possible to denoise an image by replacing the color of a pixel with the average of the colors of its neighboring pixels, this approach disregards the location of these neighboring pixels. To suppress the noise of the V channel, we decided to use the Non-Local Means Denoising algorithm [239]. This algorithm assigns a weight to each pixel based on the surrounding windows of neighboring pixels. The mathematical description of this algorithm can be found in Appendix F.1. Subsequently, the original HSV-V channel is replaced with the denoised V channel.

5.4.2 Evaluation of the filter to enhance quality of a face image

The primary objective of these experiments is to ascertain how the preprocessing filter influences the performance of face recognition. In order to assess the filter's impact on the recognition process comprehensively, it was necessary to create a custom dataset that simulates various types of shadows.

Dataset

A face image can be corrupted by various combinations of non-uniform brightness. To simulate different shadow variants, we modified the images from the LFW dataset using two different groups of masks. The first set, called *hard shadows* represents situations in which the face image is strongly affected by non-uniform illumination. The second set is distinct because it focuses on simulating conditions that may occur frequently, namely, *soft shadows*.

Figure 5.3 demonstrates an example of masks applied to images from the LFW dataset in order to simulate shadows.

Figure 5.3: Examples of masks used to simulate shadows in the LFW dataset, where the first row consists of images modified by "hard shadow masks" and the second row consists of images modified by "soft shadow masks".

Experiments

First, experiments were conducted to determine the optimal size of the Gaussian kernel. Since the algorithm is used in the face recognition system, the influence on face recognition was evaluated using the ArcFace algorithm, with the threshold determined by the highest accuracy result found on the ROC curve generated using the LFW dataset.

To evaluate the performance, we compared triplets from the dataset containing the reference, shadowed, and enhanced images. Each triplet from the dataset was converted into embedding vectors using ArcFace. The effect of the filter is then observed by comparing the L2 distances within the individual triplet. The kernel size affects both corrupted images and images with uniform illumination. Therefore, it is necessary to investigate the effect of the filter kernel size in both cases.

For practical use, it is necessary to balance these factors together with the FAR and FRR parameters. First, we deal with the filter to enhance the image so that the L2 distance between the reference and the enhanced image is below the threshold, see Table 5.13.

Table 5.13: Summary of the Gaussian kernel size to improve face recognition performance without using denoising.

Kernel size	**#images above threshold before enhancement**	**#images above threshold after enhancement**
3×3	10,077	2,660
5×5	10,077	2,840
10×10	10,077	4,264
15×15	10,077	5,637
20×20	10,077	6,611
25×25	10,077	7,260
30×30	10,076	7,902
35×35	10,077	7,643

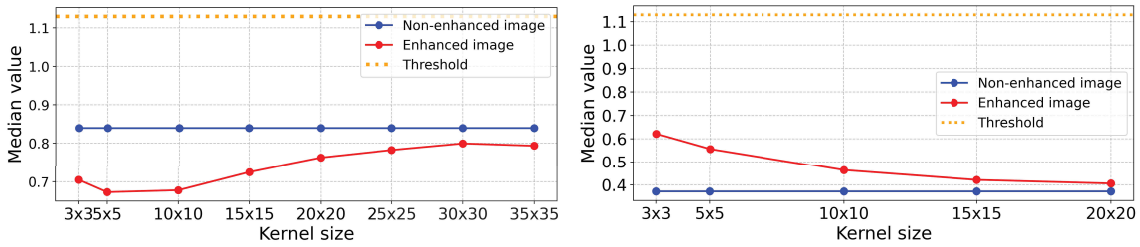
We found that using the kernel with size 3×3 servers to reduce the distance between reconstructed and reference image. Moreover, when denoising is not used, the results for 3×3 are slightly better (only 2,442 samples are above the threshold). Therefore, the experiments with the second group of damages were performed with and without denoising, see Table 5.14.

Table 5.14: Summary of the Gaussian kernel size to improve face recognition performance. The results are obtained using ArcFace on the dataset with soft shadowed images.

Kernel size	**#images above threshold before enhancement**	**#images above threshold after enhancement**	**#images above threshold enhancement+denoised**
3×3	435	52	55
5×5	435	6	7
10×10	435	6	18
15×15	435	29	45
20×20	435	81	98

Since the filter affects all comparisons, it is necessary to observe changes in the medians of the distributions both before and after reconstruction. Figures 5.4(a,b) show the median curve progression for soft and hard shadows, respectively. It is evident that the application

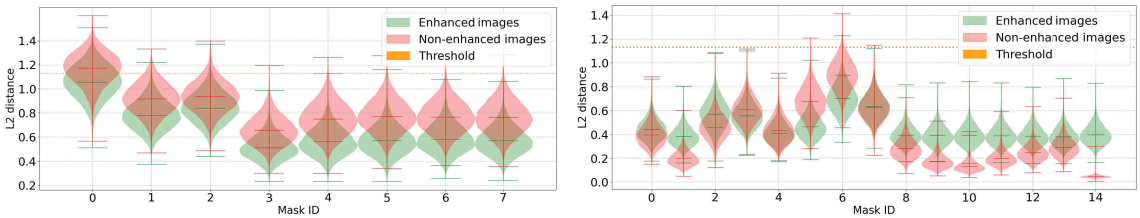
of the filter leads to a larger gap between the reconstructed image and the reference image. Nevertheless, the median is still well below the established threshold based on the EER.



(a) Hard shadows (b) Soft shadows

Figure 5.4: The median curve progression as a function of Gaussian filtering kernel size.

Furthermore, distributions of the comparisons were generated for each mask. In Figure 5.5 (a, b), where the mask ID identifies the mask applied to the face image, we can observe that the distribution of comparison distances is affected by the proposed filter in each case. However, the distances are sufficiently distant from the threshold, so while the gap between the vectors increases, the impact on recognition is not substantial. Conversely, the comparison distances exceeded the threshold before being reduced by the filter. Similarly, distributions were generated for kernels with other sizes, as shown in Appendix F.2.



(a) Hard shadows (b) Soft shadows

Figure 5.5: Distributions of L2 distances between reference image and non-enhanced/enhanced images with soft shadows for kernel 10×10.

The filter is easy to implement and use, and we recommend using it before processing the face image with the neural network for an embedded device. Overall, using the 10×10 kernel does not give the best possible results, but has less impact on shadow images. For cases where processing speed is not critical, we recommend considering the use of neural networks, which also have a positive effect on recognition, as we have shown in [TG.8].

5.5 Summary

In this chapter, we focused on face recognition on embedded devices, which is divided into detection and recognition phases. Since most modern face detectors use feature pyramid networks (FPN) to improve accuracy, we chose RetinaFace, a detector that also uses this sub-architecture. In general, backbones of neural network architectures have strong impact on their accuracy and inference speed. Likewise for RetinaFace architecture, these parameters can be modified by changing its backbone network. Our focus was mainly on

the performance of the EfficienNet family. First, we evalauted behavior of EfficienNets inferred on the GPU, where we discovered that the EfficientNetV2-S detector has better average precision than the ResNet50 backbone detector.

Regarding the embedded devices, we found that using EfficientNet-Lite B0 on Intel Neural Compute Stick 2 and Rockchip NPU accelerators results in only sligthly lower average detection precision than the originaly used ResNet50, but the processing speed is significantly better. The exception was the Hailo accelerator, which, within our observations, had the best response with the ResNet50 backbone. However, the compromise between accuracy and inference speed should be considered with respect to the intended usage of the embedded system.

Our experiments show that the models based on EfficientNet-Lite B0 and EfficientNet-Lite B1 can be successfully processed by the Rockchip NPU and the Hailo device. Using the Hailo device, face recognition can be performed in less than 10 ms. Overall, the phase responsible for mapping a 112×112 resolution face image to an embedding 512-dimensional vector is significantly faster than the face detection step, showing that the detection phase is the bottleneck of the face recognition process performed on an embedded device.

In an effort to speed up the solution, we experimented with applying a pruning algorithm to the RetinaFace Edge networks. Unfortunately, this approach did not result in any form of acceleration, primarily due to the complex nature of the RetinaFace Edge models and the substantial interdependencies within their architectures.

The final part of this chapter saw an introduction of our design for a filter for improving non-uniform brightness in face images. This filter represents a less computationally demanding alternative to enhancing the quality of face images using generative neural networks, which is feasible for devices with limited performance.

Chapter 6

Conclusion

This thesis deals with extensive research in the field of face recognition. To establish a theoretical framework, the initial part of the work identifies and summarizes the state-of-the-art approaches and methods in this field. The first research task was focused on the generation of face datasets that are reliable for evaluation of biometric systems based on face recognition. This was achieved by simulating the real conditions and preserving the facial features, which is a crucial aspect that distinguishes our dataset from those generated by AI, as we have also validated experimentally. As a part of this research task, we have examined in detail different components of the pipeline for creating a generated dataset, such as obtaining models using a 3D scanning device and an application for dataset generation. To generate a dataset tailored to contain different head poses, we used our SYDAGenerator application and the scans of 3D head models contained in the SYDA-Fidentis dataset. The result contains images of 216 subjects captured at three simulated distances with various yaw and pitch head rotations, the majority of whom are of Caucasian descent.

The verification of the credibility of the generated dataset was split into two parts. By comparing the impostor and genuine distributions of two commonly used face recognition datasets (LFW and CFP) to the distributions obtained using the generated dataset, we verified that the face recognition algorithms can discriminate between different identities represented in the generated images. However, for the second part, for verifying the behavior of detection and recognition algorithms when working on generated images of faces at angles where facial texture is evident, we were faced with a small causality dilemma. To evaluate these situations properly, we would ideally need labeled datasets containing real faces under various angles. But these are hard to come by, and are too limited in sample size for our needs, which, coincidentally, is the reason for creating our generated dataset in the first place. Nevertheless, despite the use of a small number of samples, we were able to clearly demonstrate that the behavior of the detection and recognition algorithms on generated images is comparable to recognition on datasets of real faces. Based on this comparison, we also determined the rotation range in which the dataset can be used to evaluate algorithms under various conditions, which can be simulated by SYDAGenerator.

While studying the behavior of face detection and recognition algorithms using a dataset of real images, we discovered that the pitch rotation can significantly affect recognition accuracy, which has important implications for biometrics, forensics and public security.

The next part of the thesis deals with algorithms related to face detection and recognition on embedded devices, known as edge devices in IoT terms. While experimenting with existing algorithms for the aim of designing our own approach, we mainly focused on the backbone module, which is responsible for extracting features from the input images.

For this purpose, our research efforts focused on the EfficientNet family of networks. Since a backbone module has a direct impact on the overall performance, the main features of these networks that we considered were accuracy, number of parameters, and above all, the possibility of acceleration. An important part of this research was to reduce the inference time of the algorithms while maintaining the level of accuracy using devices designed for neural network acceleration - Intel Compute Neural Stick 2, Rockchip NPU and Hailo.

The face detection phase of the pipeline for the proposed approach is based on the RetinaFace architecture. As a part of our experiments, we discovered that disregarding the original backbone options and using EfficientV2-S instead, which also has fewer parameters, offers better average precision on some datasets.

Even though the use of the neural network accelerators enabled us to overcome the performance limitations of embedded devices and opt for more computationally demanding backbones with higher levels of accuracy, the overall processing time is still significantly affected by the post-processing phase. This phase, which includes tasks such as box extraction and position computation, is the main bottleneck in acceleration of the modern face detection approaches, and is particularly challenging for embedded devices.

On the other hand, when it comes to face recognition, the process of face embedding can be accomplished within a matter of tens of milliseconds. If one is open to a slight compromise in accuracy in exchange for a fast processing time, leveraging the MagFace model with the EfficientNet-Lite B0 backbone and the Hailo accelerator can achieve face recognition under 10 milliseconds.

Face image quality has strong effect on the process of face embedding, and surveillance cameras often capture images under varying lighting conditions. In order to address the problem of shadow effects on these images, we decided to add a preprocessing step to our pipeline. While the first option to enhance the quality of the images would be to employ a neural network, this is not feasible for performance constrained embedded devices. Therefore, we designed a lightweight algorithm using HSV and Gaussian techniques and conducted a series of experiments to fine-tune parameters such as kernel size, aiming to determine the optimal configuration for improving image quality.

Overall, we explored the means of face recognition for embedded devices. We expect embedded devices, also called edge devices, will play a significant role in the new generation of surveillance systems. With well-defined regulations, standards, and rules, these systems have the potential to provide new ways of mitigating the growing privacy concerns. A system utilizing an edge device would be able to recognize a person of interest directly at the scene, discarding the face data of unrelated individuals and minimizing the need for their off-site processing.

We expect that further collaboration with police and experts in the field of anthropometry may lead to an expansion of our research and a further deployment of our application.

Author's publications

TG.1. DRAHANSKÝ, Martin. *Hand-Based Biometrics: Methods and Technology*. London, GB: The Institution of Engineering and Technology, 2018. ISBN 978-1-78561-224-4. Available also from: https://www.fit.vut.cz/research/publication/11556.

TG.2. GOLDMANN, Tomáš; DRAHANSKÝ, Martin. Generating Face Image Dataset Using a 3D Head Model. In: *2021 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*. Rajshahi, BD: Institute of Electrical and Electronics Engineers, 2021, pp. 1–4. ISBN 978-1-6654-0638-3. Available from DOI: 10.1109/IC4ME253898.2021.9768496.

TG.3. GOLDMANN, Tomáš; DRAHANSKÝ, Martin. *HeadViewer - simulation tool for viewing head from the CCTV camera position*. 2016. Available also from: https://www.fit.vut.cz/research/product/501/.en.

TG.4. DRAHANSKÝ, Martin; SPURNÝ, Martin; GOLDMANN, Tomáš. Gesichtsdetektion und -erkennung in Videos aus öffentlichen Kamerasystemen. *DuD - Datenschutz und Datensicherheit*. 2017, vol. 41, no. 7, pp. 415–421. ISSN 1614-0702. Available from DOI: 10.1007/s11623-017-0804-1.

TG.5. GOLDMANN, Tomáš; DRAHANSKÝ, Martin. *3D face scanner* [online]. 2020. [visited on 2023-08-19]. Available from: https://isdv.upv.cz/webapp/resdb.print_detail.det?pspis=PUV/37950&plang=EN. utility model.

TG.6. GOLDMANN, Tomáš; DRAHANSKÝ, Martin. *SYDAGenerator 2 - Advanced tool for generating datasets using a 3D object*. 2020. Available also from: https://www.fit.vut.cz/research/product/668/.en.

TG.7. LA, Maurizio Simone Cava; ORRU, Giulia; GOLDMANN, Tomáš; DRAHANSKÝ, Martin; MARCIALIS, Luca Gian. 3D Face Reconstruction for Forensic Recognition - A Survey. In: *2022 26th International Conference on Pattern Recognition (ICPR)*. Manhattan, New York, US, 2022, pp. 930–937. ISBN 978-1-6654-9062-7. Available from DOI: 10.1109/ICPR56361.2022.9956031.

TG.8. PLEŠKO, Filip; GOLDMANN, Tomáš; MALINKA, Kamil. Face reconstruction and its influence to face recognition. In: 2023. IEEE Xplore proceeding of *BIOSIG 2023* September 2023.

Bibliography

1. ZHANG, Shikun; FENG, Yuanyuan; SADEH, Norman. Facial recognition: Understanding privacy concerns and attitudes across increasingly diverse deployment scenarios. In: *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 2021, pp. 243–262.
2. MAYHEW, Stephen. *History of biometrics: Biometric update*. 2018. Available also from: https://www.biometricupdate.com/201802/history-of-biometrics-2.
3. FENNELLY, Lawrence J. *Effective physical security*. Butterworth-Heinemann, 2016.
4. DRAHANSKÝ, Martin; ORSÁG, Filip; DOLEŽEL, Michal. *Biometrie*. M. Drahanskỳ, 2011.
5. JAIN, Anil; HONG, Lin; PANKANTI, Sharath. Biometric identification. *Communications of the ACM*. 2000, vol. 43, no. 2, pp. 90–98.
6. SANJEKAR, PS; PATIL, JB. An overview of multimodal biometrics. *Signal & Image Processing*. 2013, vol. 4, no. 1, p. 57.
7. MIN, Rui; CHOI, Jongmoo; MEDIONI, Gérard; DUGELAY, Jean-Luc. Real-time 3D face identification from a depth camera. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 1739–1742.
8. HERNÁNDEZ ÁLVAREZ, Fernando; HERNÁNDEZ ENCINAS, Luis; SÁNCHEZ ÁVILA, Carmen. Biometric fuzzy extractor scheme for iris templates. 2009.
9. JAIN, Anil; BOLLE, Ruud; PANKANTI, Sharath. *Introduction to biometrics*. Springer, 1996.
10. ROSS, Arun; JAIN, Anil K. Multimodal biometrics: An overview. In: *2004 12th European signal processing conference*. IEEE, 2004, pp. 1221–1224.
11. PORTER, Glenn; DORAN, Greg. An anatomical and photographic technique for forensic facial identification. *Forensic science international*. 2000, vol. 114, no. 2, pp. 97–105.
12. BLEDSOE, Woodrow Wilson. The model method in facial recognition. *Panoramic Research Inc., Palo Alto, CA, Rep. PR1*. 1964, vol. 15, no. 47, p. 2.
13. TURK, Matthew A; PENTLAND, Alex P. Face recognition using eigenfaces. In: *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*. IEEE Computer Society, 1991, pp. 586–587.

14. TASKIRAN, Murat; KAHRAMAN, Nihan; ERDEM, Cigdem Eroglu. Face recognition: Past, present and future (a review). *Digital Signal Processing*. 2020, vol. 106, p. 102809.

15. MCKENNA, Stephen J; GONG, Shaogang; WÜRTZ, Rolf P; TANNER, Jonathan; BANIN, Daniel. Tracking facial feature points with Gabor wavelets and shape models. In: *Audio-and Video-based Biometric Person Authentication: First International Conference, AVBPA'97 Crans-Montana, Switzerland, March 12–14, 1997 Proceedings 1*. Springer, 1997, pp. 35–42.

16. GATES, Kelly A. *Our biometric future: Facial recognition technology and the culture of surveillance*. Vol. 2. NYU Press, 2011.

17. SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

18. KORTLI, Yassin; JRIDI, Maher; AL FALOU, Ayman; ATRI, Mohamed. Face recognition systems: A survey. *Sensors*. 2020, vol. 20, no. 2, p. 342.

19. TERHÖRST, Philipp. *Mitigating soft-biometric driven bias and privacy concerns in face recognition systems*. 2021. PhD thesis.

20. BELHUMEUR, Peter N.; HESPANHA, Joao P; KRIEGMAN, David J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*. 1997, vol. 19, no. 7, pp. 711–720.

21. DAVIES, ER. Introduction to texture analysis. In: *Handbook of texture analysis*. World Scientific, 2008, pp. 1–31.

22. AHONEN, Timo; HADID, Abdenour; PIETIKÄINEN, Matti. Face recognition with local binary patterns. In: *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part I 8*. Springer, 2004, pp. 469–481.

23. ŠTRUC, Vitomir; PAVEŠIĆ, Nikola, et al. Principal gabor filters for face recognition. In: *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*. IEEE, 2009, pp. 1–6.

24. GUPTA, Surbhi; THAKUR, Kutub; KUMAR, Munish. 2D-human face recognition using SIFT and SURF descriptors of face's feature regions. *The Visual Computer*. 2021, vol. 37, pp. 447–456.

25. CAO, Zhimin; YIN, Qi; TANG, Xiaoou; SUN, Jian. Face recognition with learning-based descriptor. In: *2010 IEEE Computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2707–2714.

26. LI, Pei; PRIETO, Loreto; MERY, Domingo; FLYNN, Patrick J. On low-resolution face recognition in the wild: Comparisons and new techniques. *IEEE Transactions on Information Forensics and Security*. 2019, vol. 14, no. 8, pp. 2000–2012.

27. BOESCH, Gaudenz. *Face Detection: Real-Time Deep Learning Applications (2023 Guide) - viso.ai — viso.ai* [online]. 2023. [visited on 2023-05-14]. Available from: https://viso.ai/deep-learning/face-detection-overview/.

28. SAHA, Priya; BHOWMIK, Mrinal Kanti; BHATTACHARJEE, Debotosh; DE, Barin Kumar; NASIPURI, Mita. Expressions Recognition of North-East Indian (NEI) Faces. *Multimedia Tools and Applications*. 2016, vol. 75, pp. 16781–16807.

29. ZHAO, He; YING, Xianghua; SHI, Yongjie; TONG, Xin; WEN, Jingsi; ZHA, Hongbin. RDCFace: radial distortion correction for face recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7721–7730.

30. HILL, Kashmir. Another arrest, and jail time, due to a bad facial recognition match. *The New York Times*. 2020, vol. 29.

31. BUOLAMWINI, Joy; GEBRU, Timnit. Gender shades: Intersectional accuracy disparities in commercial gender classification. In: *Conference on fairness, accountability and transparency*. PMLR, 2018, pp. 77–91.

32. TOVEE, MJ et al. Anthropometry. 2012.

33. SHAN, Zhiyi; HSUNG, Richard Tai-Chiu; ZHANG, Congyi; JI, Juanjuan; CHOI, Wing Shan; WANG, Wenping; YANG, Yanqi; GU, Min; KHAMBAY, Balvinder S. Anthropometric accuracy of three-dimensional average faces compared to conventional facial measurements. *Scientific Reports*. 2021, vol. 11, no. 1, p. 12254.

34. KOPECKỳ, Miroslav; KREJČOVSKỳ, Lubomır; ŠVARC, Marek. *Anthropometric measuring tools and methodology for the measurement of anthropometric parameters*. Palackỳ University, 2014.

35. IR, Farkas LG Munro. Anthropometric facial proportion in medicine. *Springfield, IL, Charles C Thomas Publisher*. 1987.

36. LEE, Wonsup; LEE, Baekhee; YANG, Xiaopeng; JUNG, Hayoung; BOK, Ilgeun; KIM, Chulwoo; KWON, Ochae; YOU, Heecheon. A 3D anthropometric sizing analysis system based on North American CAESAR 3D scan data for design of head wearable products. *Computers & Industrial Engineering*. 2018, vol. 117, pp. 121–130.

37. LORKIEWICZ-MUSZYŃSKA, Dorota; KOCIEMBA, Wojciech; ŻABA, Czesław; ŁABĘCKA, Marzena; KORALEWSKA-KORDEL, Małgorzata; ABREU-GŁOWACKA, Monica; PRZYSTAŃSKA, Agnieszka. The conclusive role of postmortem computed tomography (CT) of the skull and computer-assisted superimposition in identification of an unknown body. *International Journal of Legal Medicine*. 2013, vol. 127, pp. 653–660.

38. OUANAN, Hamid; OUANAN, Mohammed; AKSASSE, Brahim. Facial landmark localization: Past, present and future. In: *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*. IEEE, 2016, pp. 487–493.

39. TORRES, Helena R; MORAIS, Pedro; FRITZE, Anne; OLIVEIRA, Bruno; VELOSO, Fernando; RÜDIGER, Mario; FONSECA, Jaime C; VILAÇA, João L. 3D Facial Landmark Localization for cephalometric analysis. In: *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2022, pp. 1016–1019.

40. CAPLE, Jodi; STEPHAN, Carl N. A standardized nomenclature for craniofacial and facial anthropometry. *International journal of legal medicine*. 2016, vol. 130, no. 3, pp. 863–879.

41. DJORDJEVIC, Jelena; ZHUROV, Alexei I; RICHMOND, Stephen; CONSORTIUM, Visigen. Genetic and environmental contributions to facial morphological variation: a 3D population-based twin study. *PloS one*. 2016, vol. 11, no. 9, e0162250.

42. RIZWAN, Syeda Amna; JALAL, Ahmad; GOCHOO, Munkhjargal; KIM, Kibum. Robust active shape model via hierarchical feature extraction with SFS-optimized convolution neural network for invariant human age classification. *Electronics*. 2021, vol. 10, no. 4, p. 465.

43. JILANI, Shelina Khalid; UGAIL, Hassan; LOGAN, Andrew. Inter-Ethnic and Demic-Group variations in craniofacial anthropometry: a review. *PSM Biological Research*. 2018, vol. 4, no. 1, pp. 6–16.

44. PANDEY, Niraj; GOGOI, Parmananda; BUDATHOKI, Deepesh; GOPAL, KC. Anthropometric study of facial index of medical students. *Journal of Kathmandu Medical College*. 2015, vol. 4, no. 4, pp. 131–134.

45. GUPTA, Shalini; MARKEY, Mia K; BOVIK, Alan C. Anthropometric 3D face recognition. *International journal of computer vision*. 2010, vol. 90, pp. 331–349.

46. MHALENI, Vutlhari C; MAPONYA, Maserebane B; RAMAKATSA, Lebohang ND; MAHLAKWANA, Lerato; MATHEBULA, Solani D. Interpupillary distance measurements for the African population of Polokwane in Limpopo province, South Africa. *African Vision and Eye Health*. 2021, vol. 80, no. 1, p. 5.

47. GORDON, Claire C; CHURCHILL, Thomas; CLAUSER, Charles E; BRADTMILLER, Bruce; MCCONVILLE, John T, et al. Anthropometric survey of US army personnel: methods and summary statistics 1988. *DTIC Document*. 1989.

48. HARMON, Leon D. The recognition of faces. *Scientific American*. 1973, vol. 229, no. 5, pp. 70–83.

49. MURRAY, Nicholas P; HUNFALVAY, Melissa; BOLTE, Takumi. The reliability, validity, and normative data of interpupillary distance and pupil diameter using eye-tracking technology. *Translational vision science & technology*. 2017, vol. 6, no. 4, pp. 2–2.

50. CAPON, Thomas. Standardised anatomical alignment of the head in a clinical photography studio. A comparison between the Frankfort Horizontal and the natural head position. *Journal of Visual Communication in Medicine*. 2016, vol. 39, no. 3-4, pp. 105–111.

51. MOBILIO, Giuseppe et al. Your face is not new to me–Regulating the surveillance power of facial recognition technologies. *Internet Policy Review*. 2023, vol. 12, pp. 1–31.

52. L'EUROPE, Conseil de et al. *Convention for the protection of individuals with regard to automatic processing of personal data*. Vol. 108. Council of Europe, 1981.

53. DE TERWANGNE, Cecile. Council of Europe convention 108+: A modernised international treaty for the protection of personal data. *Computer Law & Security Review*. 2021, vol. 40, p. 105497.

54. COUNCIL OF EUROPE. *Guidelines on facial recognition* [online]. Council of Europe, 2021 [visited on 2023-07-16]. Available from: https://edoc.coe.int/en/artificial-intelligence/9753-guidelines-on-facial-recognition.html#.

55. EUROPEAN PARLIAMENT; COUNCIL OF THE EUROPEAN UNION. *Regulation (EU) 2016/679 of the European Parliament and of the Council* [of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)] [online]. OJ L 119, 4.5.2016, p. 1–88, 2016-05-04 [visited on 2023-04-13]. Available from: https://data.europa.eu/eli/reg/2016/679/oj.

56. LAW, Justia. *Biometric Information Privacy Act*. 2013. Available also from: https://law.justia.com/codes/illinois/2013/chapter-740/act-740-ilcs-14/.

57. VIOLA, Paul; JONES, Michael. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. IEEE, 2001, vol. 1, pp. I–I.

58. WITTEK, Peter. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

59. CHEN, Jie; CHENG, Sheng; XU, Meng. A Face Detection Method Based on Sliding Window and Support Vector Machine. *J. Comput.* 2019, vol. 14, no. 7, pp. 470–478.

60. DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Ieee, 2005, vol. 1, pp. 886–893.

61. SURASAK, Thattapon; TAKAHIRO, Ito; CHENG, Cheng-hsuan; WANG, Chi-en; SHENG, Pao-you. Histogram of oriented gradients for human detection in video. In: *2018 5th International conference on business and industrial research (ICBIR)*. IEEE, 2018, pp. 172–176.

62. HUANG, Kevin. Principal Component Analysis in the Eigenface Technique for Facial Recognition. *Trinity College Digital Repository*. 2012. Available also from: https://digitalrepository.trincoll.edu/theses/216.

63. FISHER, Ronald A. The statistical utilization of multiple measurements. *Annals of eugenics*. 1938, vol. 8, no. 4, pp. 376–386.

64. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Biometrics*. Geneva, CH, 2023. Standard, ISO/IEC JTC 1/SC 37. International Organization for Standardization. Available also from: https://www.iso.org/committee/313770.html.

65. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Cards and security devices for personal identification*. Geneva, CH, 2023. Standard, ISO/IEC JTC 1/SC 17. International Organization for Standardization. Available also from: https://www.iso.org/committee/45144.html.

66. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information security, cybersecurity and privacy protection*. Geneva, CH, 2023. Standard, ISO/IEC JTC 1/SC 27. International Organization for Standardization. Available also from: https://www.iso.org/committee/45306.html.

67. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Biometric application programming interface — Part 1: BioAPI specification*. Geneva, CH, 2018. Standard, ISO/IEC 19784-1:2018. International Organization for Standardization. Available also from: https://www.iso.org/standard/70866.html.

68. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Biometrics — BioAPI Interworking Protocol*. Geneva, CH, 2018. Standard, ISO/IEC 24708:2008. International Organization for Standardization. Available also from: https://www.iso.org/standard/43611.html.

69. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Biometric data interchange formats — Part 5: Face image data*. Geneva, CH, 2020. Standard, ISO/IEC 19785-1:2020. International Organization for Standardization. Available also from: https://www.iso.org/standard/77892.html.

70. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Biometric data interchange formats — Part 5: Face image data*. Geneva, CH, 2011. Standard, ISO/IEC 19794-5:2011. International Organization for Standardization. Available also from: https://www.iso.org/standard/50867.html.

71. FIDO ALLIANCE. *Biometric Component Certification* [online]. 2023. [visited on 2023-01-16]. Available from: https://fidoalliance.org/certification/biometric-component-certification/.

72. ANDROID OPEN SOURCE PROJECT. *Android Open Source Project* [online]. 2023. [visited on 2023-01-16]. Available from: https://source.android.com/docs/security/features/biometric.

73. WOLF, Andreas. Portrait quality (reference facial images for mrtd). *Version: 0.06 ICAO, Published by authority of the Secretary General*. 2016, vol. 3.

74. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*. Geneva, CH, 1994. Standard, ISO/IEC 10918-1:1994. International Organization for Standardization. Available also from: https://www.iso.org/standard/18902.html.

75. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*. Geneva, CH, 2019. Standard, ISO/IEC 15444-1:2019. International Organization for Standardization. Available also from: https://www.iso.org/standard/78321.html.

76. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification*. Geneva, CH, 2004. Standard, ISO/IEC 15948:2004. International Organization for Standardization. Available also from: https://www.iso.org/standard/29581.html.

77. INTERNATIONAL CIVIL AVIATION ORGANIZATION. *Machine Readable Travel Documents*. 2021. No. Doc 9303. ISBN 978-92-9265-333-0. Available also from: https://www.iso.org/standard/29581.html.

78. POWERS, David MW. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*. 2020.

79. BERSIMIS, F; VARLAMIS, Iraklis; VAMVAKARI, Malvina; PANAGIOTAKOS, D. *Calculation Methods for Binary Classification based on Discrete Data. An Application on Synthetic and Real Data.* 2018.

80. RODRIGUEZ-HERNÁNDEZ, MM; PRUNEDA, RE; RODRIGUEZ-DIAZ, JM. Statistical Analysis of the Evolutive Effects of Language Development in the Resolution of Mathematical Problems in Primary School Education. *Mathematics*. 2021, vol. 9, no. 10, p. 1081.

81. TASKIRAN, Murat; KAHRAMAN, Nihan; ERDEM, Cigdem Eroglu. Face recognition: Past, present and future (a review). *Digital Signal Processing*. 2020, vol. 106, p. 102809.

82. KHAN, Hassan; HENGARTNER, Urs; VOGEL, Daniel. Targeted mimicry attacks on touch input based implicit authentication schemes. In: *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 2016, pp. 387–398.

83. GOOGLE MACHINE LEARNING EDUCATION. *Classification: ROC Curve and AUC* [online]. 2022. [visited on 2023-08-03]. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

84. PADILLA, Rafael; NETTO, Sergio L; DA SILVA, Eduardo AB. A survey on performance metrics for object-detection algorithms. In: *2020 international conference on systems, signals and image processing (IWSSIP)*. IEEE, 2020, pp. 237–242.

85. DRAELOS, Rachel. *2020 international conference on systems, signals and image processing (IWSSIP)*. The Complete Guide to AUC and Average Precision: Simulations and Visualizations [online]. 2020. [visited on 2023-08-19].

86. LI, Stan Z. *Encyclopedia of Biometrics: I-Z*. Vol. 1. Springer Science & Business Media, 2009.

87. TAIGMAN, Yaniv; YANG, Ming; RANZATO, MarcÁurelio; WOLF, Lior. Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

88. MONTESINOS LÓPEZ, Osval Antonio; MONTESINOS LÓPEZ, Abelardo; CROSSA, Jose. Fundamentals of Artificial Neural Networks and Deep Learning. In: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer, 2022, pp. 379–425.

89. SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. *Neural networks*. 2015, vol. 61, pp. 85–117.

90. MEDSKER, Larry R; JAIN, LC. Recurrent neural networks. *Design and Applications*. 2001, vol. 5, no. 64-67, p. 2.

91. LU, Yulong; LU, Jianfeng. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*. 2020, vol. 33, pp. 3094–3105.

92. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

93. KANG, Kai. Comparison of face recognition and detection models: Using different convolution neural networks. *Optical Memory and Neural Networks*. 2019, vol. 28, pp. 101–108.

94. HANIN, Boris. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in neural information processing systems*. 2018, vol. 31.

95. LU, Lu; SHIN, Yeonjong; SU, Yanhui; KARNIADAKIS, George Em. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*. 2019.

96. BARUAH, Indraneel Dutta. *Activation functions and loss functions for neural networks-how to pick the right one?* [online]. Analytics Vidhya, 2021 [visited on 2022-10-06]. Available from: https://medium.com/analytics-vidhya/activation-functions-and-loss-functions-for-neural-networks-how-to-pick-the-right-one-542e1dd523e0.

97. MONTESINOS LÓPEZ, Osval Antonio; MONTESINOS LÓPEZ, Abelardo; CROSSA, Jose. Convolutional Neural Networks. In: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer, 2022, pp. 533–577.

98. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

99. AGGARWAL, Charu C et al. Neural networks and deep learning. *Springer*. 2018, vol. 10, no. 978, p. 3.

100. SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*. 2014, vol. 15, no. 1, pp. 1929–1958.

101. JANOCHA, Katarzyna; CZARNECKI, Wojciech Marian. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*. 2017.

102. NNAMOKO, Nonso; BARROWCLOUGH, Joseph; PROCTER, Jack. Solid waste image classification using deep convolutional neural network. *Infrastructures*. 2022, vol. 7, no. 4, p. 47.

103. GÓMEZ, Raúl. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names* [online]. 2018. [visited on 2022-10-06]. Available from: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

104. PONCE, Pedro; MATA, Omar; PEREZ, Esteban; LOPEZ, Juan Roberto; MOLINA, Arturo; MCDANIEL, Troy. S4 features and artificial intelligence for designing a robot against COVID-19—Robocov. *Future Internet*. 2022, vol. 14, no. 1, p. 22.

105. CARVALHO, Marcela; LE SAUX, Bertrand; TROUVÉ-PELOUX, Pauline; ALMANSA, Andrés; CHAMPAGNAT, Frédéric. On regression losses for deep depth estimation. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2915–2919.

106. ZAHEER, Raniah; SHAZIYA, Humera. A study of the optimization algorithms in deep learning. In: *2019 third international conference on inventive systems and control (ICISC)*. IEEE, 2019, pp. 536–539.

107. SUN, Shiliang; CAO, Zehui; ZHU, Han; ZHAO, Jing. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*. 2019, vol. 50, no. 8, pp. 3668–3681.

108. GENC, Burkay; TUNC, HÜSEYİN. Optimal training and test sets design for machine learning. *Turkish Journal of Electrical Engineering and Computer Sciences*. 2019, vol. 27, no. 2, pp. 1534–1545.

109. MURAINA, Ismail. Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts. In: *7th International Mardin Artuklu Scientific Research Conference*. 2022.

110. SHAHINFAR, Saleh; MEEK, Paul; FALZON, Greg. "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics*. 2020, vol. 57, p. 101085. ISSN 1574-9541. Available from DOI: https://doi.org/10.1016/j.ecoinf.2020.101085.

111. SHORTEN, Connor; KHOSHGOFTAAR, Taghi M. A survey on image data augmentation for deep learning. *Journal of big data*. 2019, vol. 6, no. 1, pp. 1–48.

112. PARKHI, Omkar M; VEDALDI, Andrea; ZISSERMAN, Andrew. Deep face recognition. 2015.

113. ZHU, Xiangxin; RAMANAN, Deva. Face detection, pose estimation, and landmark localization in the wild. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2879–2886.

114. JAIN, Vidit; LEARNED-MILLER, Erik. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. 2010. Tech. rep., UM-CS-2010-009. University of Massachusetts, Amherst.

115. YANG, Bin; YAN, Junjie; LEI, Zhen; LI, Stan Z. Fine-grained Evaluation on Face Detection in the Wild. In: *Automatic Face and Gesture Recognition (FG), 11th IEEE International Conference on*. IEEE, 2015.

116. YANG, Shuo; LUO, Ping; LOY, Chen-Change; TANG, Xiaoou. Wider face: A face detection benchmark. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5525–5533.

117. NAPHADE, Milind; SMITH, John R; TESIC, Jelena; CHANG, Shih-Fu; HSU, Winston; KENNEDY, Lyndon; HAUPTMANN, Alexander; CURTIS, Jon. Large-scale concept ontology for multimedia. *IEEE multimedia*. 2006, vol. 13, no. 3, pp. 86–91.

118. KOESTINGER, Martin; WOHLHART, Paul; ROTH, Peter M; BISCHOF, Horst. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE, 2011, pp. 2144–2151.

119. BURGOS-ARTIZZU, Xavier P; PERONA, Pietro; DOLLÁR, Piotr. Robust face landmark estimation under occlusion. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1513–1520.

120. SAGONAS, Christos; ANTONAKOS, Epameinondas; TZIMIROPOULOS, Georgios; ZAFEIRIOU, Stefanos; PANTIC, Maja. 300 faces in-the-wild challenge: Database and results. *Image and vision computing*. 2016, vol. 47, pp. 3–18.

121. WU, Wayne; QIAN, Chen; YANG, Shuo; WANG, Quan; CAI, Yici; ZHOU, Qiang. Look at Boundary: A Boundary-Aware Face Alignment Algorithm. In: *CVPR*. 2018.

122. HUANG, Gary B.; RAMESH, Manu; BERG, Tamara; LEARNED-MILLER, Erik. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. 2007-10. Tech. rep., 07-49. University of Massachusetts, Amherst.

123. ZHENG, Tianyue; DENG, Weihong; HU, Jiani. Cross-Age LFW: A Database for Studying Cross-Age Face Recognition in Unconstrained Environments. *CoRR*. 2017, vol. abs/1708.08197. Available from arXiv: 1708.08197.

124. WOLF, Lior; HASSNER, Tal; MAOZ, Itay. Face recognition in unconstrained videos with matched background similarity. In: *CVPR 2011*. IEEE, 2011, pp. 529–534.

125. MOSCHOGLOU, Stylianos; PAPAIOANNOU, Athanasios; SAGONAS, Christos; DENG, Jiankang; KOTSIA, Irene; ZAFEIRIOU, Stefanos. Agedb: the first manually collected, in-the-wild age database. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*. 2017, vol. 2, p. 5. No. 3.

126. GUO, Yandong; ZHANG, Lei; HU, Yuxiao; HE, Xiaodong; GAO, Jianfeng. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In: *ECCV 2016*. ECCV 2016. 2016. Available also from: https://www.microsoft.com/en-us/research/publication/ms-celeb-1m-dataset-benchmark-large-scale-face-recognition-2/.

127. PELLISSIER TANON, Thomas; VRANDEČIĆ, Denny; SCHAFFERT, Sebastian; STEINER, Thomas; PINTSCHER, Lydia. From freebase to wikidata: The great migration. In: *Proceedings of the 25th international conference on world wide web*. 2016, pp. 1419–1428.

128. GUO, Yandong; ZHANG, Lei; HU, Yuxiao; HE, X.; GAO, Jianfeng. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In: *ECCV*. 2016.

129. JIN, Chi; JIN, Ruochun; CHEN, Kai; DOU, Yong. A community detection approach to cleaning extremely large face database. *Computational intelligence and neuroscience*. 2018, vol. 2018.

130. YI, Dong; LEI, Zhen; LIAO, Shengcai; LI, Stan Z. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*. 2014.

131. BAE, Gwangbin; LA GORCE, Martin de; BALTRUŠAITIS, Tadas; HEWITT, Charlie; CHEN, Dong; VALENTIN, Julien; CIPOLLA, Roberto; SHEN, Jingjing. DigiFace-1M: 1 Million Digital Face Images for Face Recognition. In: *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2023.

132. KLARE, Brendan F; KLEIN, Ben; TABORSKY, Emma; BLANTON, Austin; CHENEY, Jordan; ALLEN, Kristen; GROTHER, Patrick; MAH, Alan; JAIN, Anil K. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1931–1939.

133. WHITELAM, Cameron; TABORSKY, Emma; BLANTON, Austin; MAZE, Brianna; ADAMS, Jocelyn; MILLER, Tim; KALKA, Nathan; JAIN, Anil K; DUNCAN, James A; ALLEN, Kristen, et al. Iarpa janus benchmark-b face dataset. In: *proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 90–98.

134. MAZE, Brianna; ADAMS, Jocelyn; DUNCAN, James A; KALKA, Nathan; MILLER, Tim; OTTO, Charles; JAIN, Anil K; NIGGEL, W Tyler; ANDERSON, Janet; CHENEY, Jordan, et al. Iarpa janus benchmark-c: Face dataset and protocol. In: *2018 international conference on biometrics (ICB)*. IEEE, 2018, pp. 158–165.

135. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Computer Security Resource Center* [online]. 2023. [visited on 2023-07-25]. Available from: https://csrc.nist.gov/glossary/term/government_off_the_shelf.

136. GROSS, Ralph; MATTHEWS, Iain; COHN, Jeffrey; KANADE, Takeo; BAKER, Simon. Multi-pie. *Image and vision computing*. 2010, vol. 28, no. 5, pp. 807–813.

137. SENGUPTA, Soumyadip; CHEN, Jun-Cheng; CASTILLO, Carlos; PATEL, Vishal M; CHELLAPPA, Rama; JACOBS, David W. Frontal to profile face verification in the wild. In: *2016 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2016, pp. 1–9.

138. GOURIER, Nicolas. Estimating face orientation from robust detection of salient facial features. In: *Proceedings of Pointing 2004, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK*. 2004.

139. BANSAL, Ankan; NANDURI, Anirudh; CASTILLO, Carlos D; RANJAN, Rajeev; CHELLAPPA, Rama. UMDFaces: An Annotated Face Dataset for Training Deep Networks. *arXiv preprint arXiv:1611.01484v2*. 2016.

140. ELHARROUSS, Omar; AKBARI, Younes; ALMAADEED, Noor; AL-MAADEED, Somaya. Backbones-review: Feature extraction networks for deep learning and deep reinforcement learning approaches. *arXiv preprint arXiv:2206.08016*. 2022.

141. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, vol. 25.

142. BENALI AMJOUD, Ayoub; AMROUCH, Mustapha. Convolutional neural networks backbones for object detection. In: *Image and Signal Processing: 9th International Conference, ICISP 2020, Marrakesh, Morocco, June 4–6, 2020, Proceedings 9*. Springer, 2020, pp. 282–289.

143. SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

144. HUANG, Gao; LIU, Zhuang; VAN DER MAATEN, Laurens; WEINBERGER, Kilian Q. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

145. SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

146. TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

147. SANDLER, Mark; HOWARD, Andrew; ZHU, Menglong; ZHMOGINOV, Andrey; CHEN, Liang-Chieh. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

148. HOWARD, Andrew G; ZHU, Menglong; CHEN, Bo; KALENICHENKO, Dmitry; WANG, Weijun; WEYAND, Tobias; ANDREETTO, Marco; ADAM, Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. 2017.

149. CHIU, Yu-Chen; TSAI, Chi-Yi; RUAN, Mind-Da; SHEN, Guan-Yu; LEE, Tsu-Tian. Mobilenet-SSDv2: An improved object detection model for embedded systems. In: *2020 International conference on system science and engineering (ICSSE)*. IEEE, 2020, pp. 1–5.

150. JOUPPI, Norman P; YOUNG, Cliff; PATIL, Nishant; PATTERSON, David; AGRAWAL, Gaurav; BAJWA, Raminder; BATES, Sarah; BHATIA, Suresh; BODEN, Nan; BORCHERS, Al, et al. In-datacenter performance analysis of a tensor processing unit. In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.

151. LIU, Renjie. *Higher accuracy on vision models with EfficientNet-Lite*. 2020. Available also from: https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html.

152. REZATOFIGHI, Hamid; TSOI, Nathan; GWAK, JunYoung; SADEGHIAN, Amir; REID, Ian; SAVARESE, Silvio. Generalized intersection over union: A metric and a loss for bounding box regression. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 658–666.

153. BODLA, Navaneeth; SINGH, Bharat; CHELLAPPA, Rama; DAVIS, Larry S. Soft-NMS–improving object detection with one line of code. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5561–5569.

154. DENG, J; GUO, J; ZHOU, Y; YU, J; KOTSIA, I; ZAFEIRIOU, S. Retinaface: Single-stage dense face localisation in the wild. arXiv 2019. *arXiv preprint arXiv:1905.00641*. 1905.

155. ZHU, Yanjia; CAI, Hongxiang; ZHANG, Shuhan; WANG, Chenhao; XIONG, Yichao. Tinaface: Strong but simple baseline for face detection. *arXiv preprint arXiv:2011.13183*. 2020.

156. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

157. GIRSHICK, Ross. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

158. UIJLINGS, Jasper RR; VAN DE SANDE, Koen EA; GEVERS, Theo; SMEULDERS, Arnold WM. Selective search for object recognition. *International journal of computer vision*. 2013, vol. 104, pp. 154–171.

159. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*. 2015, vol. 28.

160. LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. Ssd: Single shot multibox detector. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

161. ZHANG, Kaipeng; ZHANG, Zhanpeng; LI, Zhifeng; QIAO, Yu. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*. 2016, vol. 23, no. 10, pp. 1499–1503.

162. LI, Haoxiang; LIN, Zhe; SHEN, Xiaohui; BRANDT, Jonathan; HUA, Gang. A convolutional neural network cascade for face detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5325–5334.

163. REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

164. REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

165. TERVEN, J; CORDOVA-ESPARZA, DM. A Comprehensive Review of YOLO: From YOLOv1 and Beyond. arXiv 2023. *arXiv preprint arXiv:2304.00501*. [N.d.].

166. BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. 2020.

167. REDMON, Joseph; FARHADI, Ali. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. 2018.

168. WANG, Chien-Yao; BOCHKOVSKIY, Alexey; LIAO, Hong-Yuan Mark. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7464–7475.

169. SOUDY, Mohamed; AFIFY, Yasmine; BADR, Nagwa. RepConv: A novel architecture for image scene classification on Intel scenes dataset. *International Journal of Intelligent Computing and Information Sciences*. 2022, vol. 22, no. 2, pp. 63–73.

170. WANG, Chien-Yao; YEH, I-Hau; LIAO, Hong-Yuan Mark. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*. 2021.

171. LIN, Tsung-Yi; GOYAL, Priya; GIRSHICK, Ross; HE, Kaiming; DOLLÁR, Piotr. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

172. LIN, Tsung-Yi; DOLLÁR, Piotr; GIRSHICK, Ross; HE, Kaiming; HARIHARAN, Bharath; BELONGIE, Serge. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

173. GUO, Jia; DENG, Jiankang; LATTAS, Alexandros; ZAFEIRIOU, Stefanos. Sample and computation redistribution for efficient face detection. *arXiv preprint arXiv:2105.04714*. 2021.

174. RADOSAVOVIC, Ilija; KOSARAJU, Raj Prateek; GIRSHICK, Ross; HE, Kaiming; DOLLÁR, Piotr. Designing network design spaces. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10428–10436.

175. INSALL, Matt; ROWLAND, Todd; WEISSTEIN, Eric W. *"Embedding." From MathWorld–A Wolfram Web Resource* [online]. 2022. [visited on 2022-02-20]. Available from: https://mathworld.wolfram.com/Embedding.html.

176. ABDELWAHAB, Ahmed; LANDWEHR, Niels. Deep Distributional Sequence Embeddings Based on a Wasserstein Loss. *arXiv preprint arXiv:1912.01933*. 2019.

177. MAHARANI, Devira Anggi; MACHBUB, Carmadi; RUSMIN, Pranoto Hidaya; YULIANTI, Lenni. Improving the capability of real-time face masked recognition using cosine distance. In: *2020 6th International conference on interactive digital media (ICIDM)*. IEEE, 2020, pp. 1–6.

178. TABAK, John. *Geometry: the language of space and form*. Infobase Publishing, 2014.

179. IRIANANDA, Syahroni Wahyu. Measure the Similarity of Complaint Document Using Cosine Similarity Based on Class-Based Indexing. *International Journal of Computer Applications Technology and Research*. 2018, vol. 7, no. 08, pp. 292–296.

180. CIAPARRONE, Gioele; CHIARIGLIONE, Leonardo; TAGLIAFERRI, Roberto. A comparison of deep learning models for end-to-end face-based video retrieval in unconstrained videos. *Neural Computing and Applications*. 2022, vol. 34, no. 10, pp. 7489–7506.

181. *Papers with code - labeled faces in the wild benchmark (face verification)* [online]. [N.d.]. [visited on 2023-02-12]. Available from: https://paperswithcode.com/sota/face-verification-on-labeled-faces-in-the.

182. NWANKPA, Chigozie; IJOMAH, Winifred; GACHAGAN, Anthony; MARSHALL, Stephen. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*. 2018.

183. WEN, Yandong; ZHANG, Kaipeng; LI, Zhifeng; QIAO, Yu. A discriminative feature learning approach for deep face recognition. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 499–515.

184. LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, vol. 86, no. 11, pp. 2278–2324.

185. CHEN, Bor-Chun; CHEN, Chu-Song; HSU, Winston H. Cross-Age Reference Coding for Age-Invariant Face Recognition and Retrieval. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

186. LIU, Ziwei; LUO, Ping; WANG, Xiaogang; TANG, Xiaoou. Deep Learning Face Attributes in the Wild. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

187. LIU, Weiyang; WEN, Yandong; YU, Zhiding; LI, Ming; RAJ, Bhiksha; SONG, Le. Sphereface: Deep hypersphere embedding for face recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 212–220.

188. WANG, Hao; WANG, Yitong; ZHOU, Zheng; JI, Xing; GONG, Dihong; ZHOU, Jingchao; LI, Zhifeng; LIU, Wei. Cosface: Large margin cosine loss for deep face recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5265–5274.

189. DENG, Jiankang; GUO, Jia; XUE, Niannan; ZAFEIRIOU, Stefanos. Arcface: Additive angular margin loss for deep face recognition. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4690–4699.

190. RANJAN, Rajeev; CASTILLO, Carlos D; CHELLAPPA, Rama. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*. 2017.

191. MENG, Qiang; ZHAO, Shichao; HUANG, Zhida; ZHOU, Feng. Magface: A universal representation for face recognition and quality assessment. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14225–14234.

192. TERHÖRST, Philipp; IHLEFELD, Malte; HUBER, Marco; DAMER, Naser; KIRCHBUCHNER, Florian; RAJA, Kiran; KUIJPER, Arjan. Qmagface: Simple and accurate quality-aware face recognition. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 3484–3494.

193. DUTA, Ionut Cosmin; LIU, Li; ZHU, Fan; SHAO, Ling. Improved Residual Networks for Image and Video Recognition. *arXiv preprint arXiv:2004.04989*. 2020.

194. ESMAEILZADEH, Hadi; SAMPSON, Adrian; CEZE, Luis; BURGER, Doug. Neural acceleration for general-purpose approximate programs. In: *2012 45th annual IEEE/ACM international symposium on microarchitecture*. IEEE, 2012, pp. 449–460.

195. GHOLAMI, Amir; KIM, Sehoon; DONG, Zhen; YAO, Zhewei; MAHONEY, Michael W; KEUTZER, Kurt. A survey of quantization methods for efficient neural network inference. 2022, pp. 291–326.

196. COMMITTEE, Microprocessor Standards et al. *754-2019-IEEE Standard for Floating-Point Arithmetic*. IEEE, 2019.

197. CHAO, Shih-Kang; WANG, Zhanyu; XING, Yue; CHENG, Guang. Directional pruning of deep neural networks. *Advances in neural information processing systems*. 2020, vol. 33, pp. 13986–13998.

198. LIEBENWEIN, Lucas; BAYKAL, Cenk; CARTER, Brandon; GIFFORD, David; RUS, Daniela. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems*. 2021, vol. 3, pp. 93–138.

199. NAGEL, Markus; BAALEN, Mart van; BLANKEVOORT, Tijmen; WELLING, Max. Data-free quantization through weight equalization and bias correction. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1325–1334.

200. CHEN, Yiran; XIE, Yuan; SONG, Linghao; CHEN, Fan; TANG, Tianqi. A survey of accelerator architectures for deep neural networks. *Engineering*. 2020, vol. 6, no. 3, pp. 264–274.

201. SCHOR, David. *Machine Learning Processor (MLP) - Microarchitectures - ARM* [online]. 2020. [visited on 2023-07-15]. Available from: https://en.wikichip.org/wiki/arm_holdings/microarchitectures/mlp.

202. CHEN, Yiran; XIE, Yuan; SONG, Linghao; CHEN, Fan; TANG, Tianqi. A survey of accelerator architectures for deep neural networks. *Engineering*. 2020, vol. 6, no. 3, pp. 264–274.

203. INTEL. *Intel® Movidius™ Myriad™ X Vision Processing Unit* [online]. 2023. [visited on 2023-07-15]. Available from: https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu/movidius-myriad-x/products.html.

204. IONICA, Mircea Horea; GREGG, David. The movidius myriad architecture's potential for scientific computing. *IEEE Micro*. 2015, vol. 35, no. 1, pp. 6–14.

205. ROCKCHIP. *Rockchip RK3588 Datasheet*. 2021. Available also from: https://www.cnx-software.com/pdf/Rockchip%C2%A0RK3588%C2%A0Datasheet%C2%A0V0.1-20210727.pdf.

206. YU, Yu; MORA, Kenneth Alberto Funes; ODOBEZ, Jean-Marc. Robust and accurate 3d head pose estimation through 3dmm and online head model reconstruction. In: *2017 12th ieee international conference on automatic face & gesture recognition (fg 2017)*. Ieee, 2017, pp. 711–718.

207. CHENG, Shiyang; MA, Pingchuan; TZIMIROPOULOS, Georgios; PETRIDIS, Stavros; BULAT, Adrian; SHEN, Jie; PANTIC, Maja. Towards pose-invariant lip-reading. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4357–4361.

208. ZHU, Xiangyu; LIU, Xiaoming; LEI, Zhen; LI, Stan Z. Face alignment in full pose range: A 3d total solution. *IEEE transactions on pattern analysis and machine intelligence*. 2017, vol. 41, no. 1, pp. 78–92.

209. HORAUD, Radu; HANSARD, Miles; EVANGELIDIS, Georgios; MÉNIER, Clément. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine vision and applications*. 2016, vol. 27, no. 7, pp. 1005–1020.

210. NEBEL, Steve; BEEGE, Maik; SCHNEIDER, Sascha; REY, Günter Daniel. A review of photogrammetry and photorealistic 3D models in education from a psychological perspective. In: *Frontiers in education*. Frontiers Media SA, 2020, vol. 5, p. 144.

211. TOLDO, Roberto; GHERARDI, Riccardo; FARENZENA, Michela; FUSIELLO, Andrea. Hierarchical structure-and-motion recovery from uncalibrated images. *Computer Vision and Image Understanding*. 2015, vol. 140, pp. 127–143.

212. TOLDO, Roberto. Towards automatic acquisition of high-level 3D models from images. *Universita degli Studi di Verona, Verona*. 2013.

213. GHERARDI, Riccardo; FARENZENA, Michela; FUSIELLO, Andrea. Improving the efficiency of hierarchical structure-and-motion. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1594–1600.

214. GHERARDI, Riccardo; FUSIELLO, Andrea. Practical autocalibration. In: *European Conference on Computer Vision*. Springer, 2010, pp. 790–801.

215. FARENZENA, Michela; FUSIELLO, Andrea; GHERARDI, Riccardo. Structure-and-motion pipeline on a hierarchical cluster tree. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1489–1496.

216. HU, Qiong; PENG, Xi; YANG, Peng; YANG, Fei; METAXAS, Dimitris N. Robust multi-pose facial expression recognition. In: *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 1782–1787.

217. SFORZA, Chiarella; GRASSI, GianPiero; FRAGNITO, Nicola; TURCI, Michela; FERRARIO, Virgilio F. Three-dimensional analysis of active head and cervical spine range of motion: effect of age in healthy male subjects. *Clinical Biomechanics*. 2002, vol. 17, no. 8, pp. 611–614.

218. SEGURA, Carlos; HERNANDO, Javier. 3D joint speaker position and orientation tracking with particle filters. *Sensors*. 2014, vol. 14, no. 2, pp. 2259–2279.

219. PARK, Junhee; BYUN, Seong-Chan; LEE, Byung-Uk. Lens distortion correction using ideal image coordinates. *IEEE Transactions on Consumer Electronics*. 2009, vol. 55, no. 3, pp. 987–991.

220. URBANOVÁ, Petra; FERKOVÁ, Zuzana; JANDOVÁ, Marie; JURDA, Mikoláš; ČERNỳ, Dominik; SOCHOR, Jiří. *FIDENTIS 3D Face Database* [online]. 2023. [visited on 2023-02-12]. Available from: https://fidentis.cz/database.

221. CHEN, Yen-Chi. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*. 2017, vol. 1, no. 1, pp. 161–187.

222. JANTUNEN, Tommi; MESCH, Johanna; PUUPPONEN, Anna; LAAKSONEN, Jorma. On the rhythm of head movements in Finnish and Swedish Sign Language sentences. In: *Speech Prosody 8, Boston, USA, May 31-June 3, 2016*. The International Speech Communication Association (ISCA), 2016, pp. 850–853.

223. WANG, Dayong; OTTO, Charles; JAIN, Anil K. Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*. 2015.

224. RAMIS, Silvia; BUADES, Jose M; PERALES, Francisco J; MANRESA-YEE, Cristina. A novel approach to cross dataset studies in facial expression recognition. *Multimedia Tools and Applications*. 2022, vol. 81, no. 27, pp. 39507–39544.

225. ANISIMOV, Dmitriy; KHANOVA, Tatiana. Towards lightweight convolutional neural networks for object detection. In: *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2017, pp. 1–8.

226. LI, Peipei; WU, Xiang; HU, Yibo; HE, Ran; SUN, Zhenan. M2FPA: A multi-yaw multi-pitch high-quality dataset and benchmark for facial pose analysis. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 10043–10051.

227. THIESEN, Guilherme; GRIBEL, Bruno Frazão; FREITAS, Maria Perpétua Mota. Facial asymmetry: a current review. *Dental press journal of orthodontics*. 2015, vol. 20, pp. 110–125.

228. FERRRON, Emily. *There's Actually a Better Place to Put Your Home Security Cameras* [online]. 2023. [visited on 2023-08-18]. Available from: https://www.cnet.com/home/security/theres-actually-a-better-place-to-put-your-home-security-cameras/.

229. ACASANDREI, Laurentiu; BARRIGA BARROS, Ángel. FPGA implementation of an embedded face detection system based on LEON3. *FPGA implementation of an embedded face detection system based on LEON3 (2012), p 1-6*. 2012.

230. KIM, Jin Ok; KIM, Jin Soo; CHUNG, Chin Hyun. Face recognition by the LDA-Based algorithm for a video surveillance system on DSP. In: *Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part I 5*. Springer, 2005, pp. 638–646.

231. VARGHESE, Blesson; WANG, Nan; BARBHUIYA, Sakil; KILPATRICK, Peter; NIKOLOPOULOS, Dimitrios S. Challenges and opportunities in edge computing. In: *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

232. LINDNER, Tymoteusz; WYRWAŁ, Daniel; BIAŁEK, Marcin; NOWAK, Patryk. Face recognition system based on a single-board computer. In: *2020 International Conference Mechatronic Systems and Materials (MSM)*. IEEE, 2020, pp. 1–6.

233. ŠUŠTERŠIČ, Tijana; VULOVIĆ, Aleksandra; FILIPOVIĆ, Nenad; PEULIĆ, Aleksandar. FPGA implementation of face recognition algorithm. In: *Pervasive Computing Paradigms for Mental Health: Selected Papers from MindCare 2016, Fabulous 2016, and IIoT 2015 3*. Springer, 2018, pp. 93–99.

234. GHUTKE, Ragina C; NAVEEN, Ch; SATPUTE, Vishal R. A novel approach for video frame interpolation using cubic motion compensation technique. *International Journal of Applied Engineering Research*. 2016, vol. 11, no. 10, pp. 7139–7146.

235. AXIS. *https://www.axis.com/learning/web-articles/identification-and-recognition/resolution* [online]. 2023. [visited on 2023-07-12]. Available from: https://www.axis.com/learning/web-articles/identification-and-recognition/resolution.

236. CHEN, Ruoyu; CHEN, Ying. Improved Convolutional Neural Network YOLOv5 for Underwater Target Detection Based on Autonomous Underwater Helicopter. *Journal of Marine Science and Engineering*. 2023, vol. 11, no. 5, p. 989.

237. SAGE, D.; UNSER, M. Easy Java Programming for Teaching Image Processing. In: *Proceedings of the 2001 IEEE International Conference on Image Processing (ICIP'01)*. 2001, vol. III, pp. 298–301.

238. SGHAIER, Anissa; DOUIK, Ali; MACHHOUT, Mohsen, et al. FPGA implementation of filtered image using 2D Gaussian filter. *International Journal of Advanced Computer Science and Applications*. 2016, vol. 7, no. 7.

239. BUADES, Antoni; COLL, Bartomeu; MOREL, Jean-Michel. Non-local means denoising. *Image Processing On Line*. 2011, vol. 1, pp. 208–212.

Appendix A

Face standard ISO/IEC 19794-5

These figures are taken from the ISO 19794-5 [70] standard and show the recommended and inappropriate position of the face in the image.

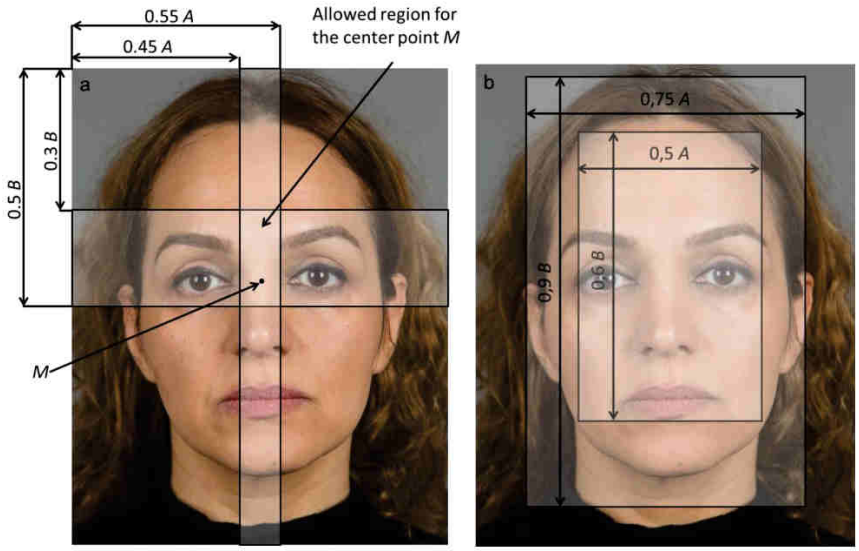


Figure A.1: Sample portraits with the respective minimal and maximal head dimensions: a) true location and allowed region for the center point M, b) minimal size (inner rectangle) and maximal size (outer rectangle, note that the position including the relative position of the two rectangles is not fixed) of image width W and head length L depending on A and B [70].

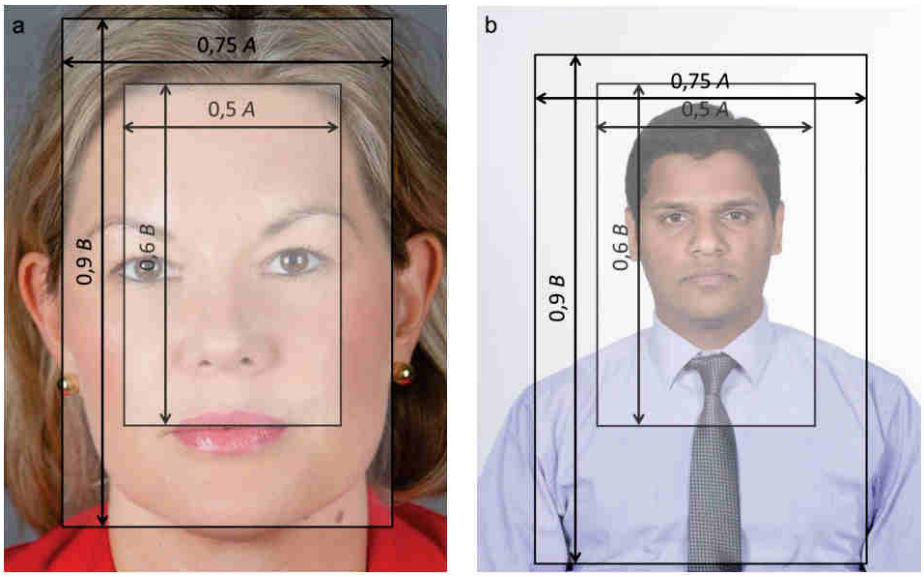


Figure A.2: Sample portraits not complying with minimal and maximal head dimensions: a) face too large and doesn't fit into the larger rectangle, b) face too small and does not cover the entire smaller rectangle [70].

Appendix B

SYDAGenerator

B.1 Preserving optical properties during image generation

To validate the plausibility of the simulation of the optical properties of the cameras, we performed an experiment in which the same properties were set in the generator as those of the reference camera. We then captured the mask of the face profile with the camera and also generated semi-synthetic images using the corresponding 3D model of the mask. Figures B.1 and B.2 show the generated image (blue background) and the mask captured by a camera.

Figure B.1: Comparison of reference (captured mask) with generated image.

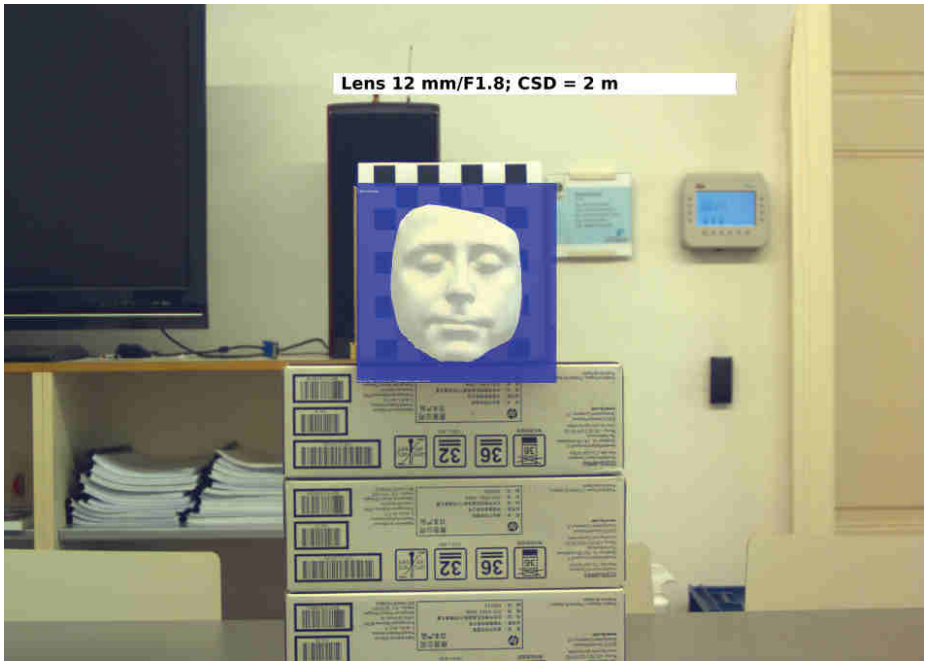
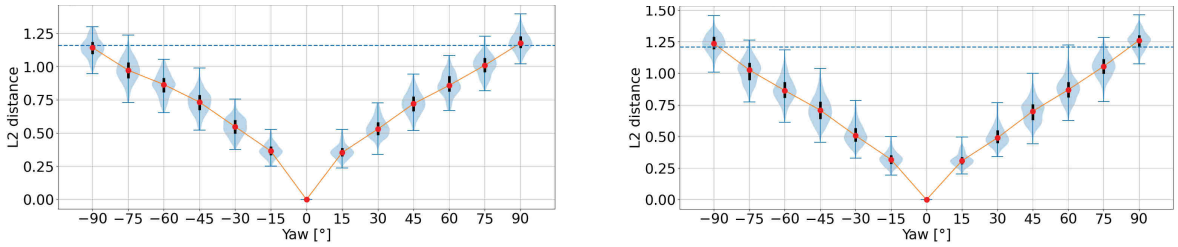


Figure B.2: Comparison of covered reference (captured mask) with generated image.

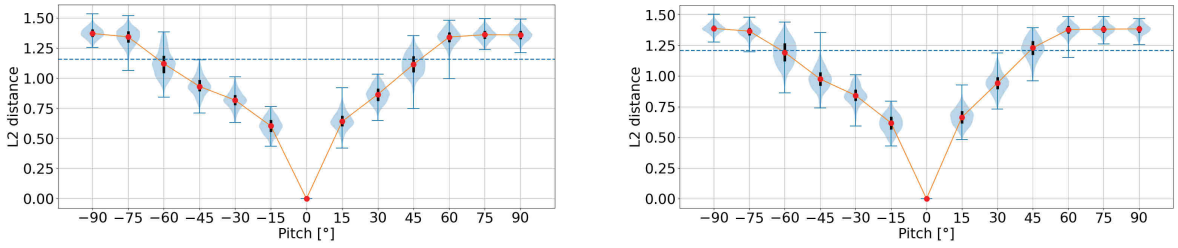
Appendix C

Influence of face rotation to performance of face recognition



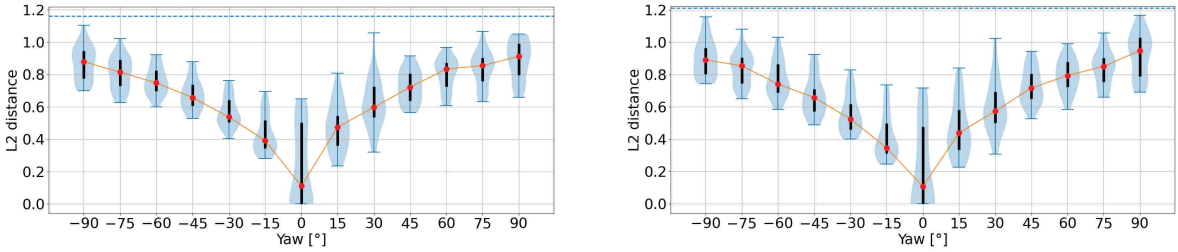
(a) (b)

Figure C.1: MagFace (a) and SphereFace (b) distributions of L2 distances for yaw-rotated faces in SYDA-Fidentis (CSD = 4 m).



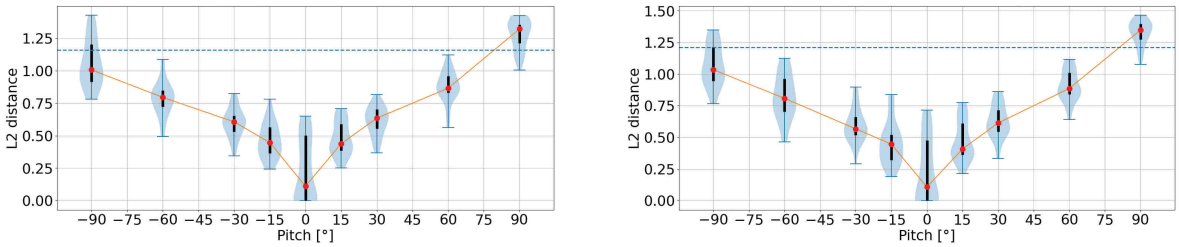
(a) (b)

Figure C.2: MagFace (a) and SphereFace (b) distributions of L2 distances for pitch-rotated faces in SYDA-Fidentis (CSD = 4 m).



(a) (b)

Figure C.3: MagFace (a) and SphereFace (b) distributions of L2 distances for yaw-rotated faces in Headpose.



(a) (b)

Figure C.4: MagFace (a) and SphereFace (b) distributions of L2 distances for pitch-rotated faces in Headpose.

Appendix D

Face detection

D.1 RetinaFace Edge - architectures

In this section we show where the RetinaFace FPN are attached to the backbones.

Table D.1: EfficientNet B0 - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	MBConv1, k3x3	16	1	1
3	MBConv6, k3x3	24	2	2
4	MBConv6, k5x5	40	2	2
5	MBConv6, k3x3	80	3	2
6	MBConv6, k5x5	112	3	1
7	MBConv6, k5x5	192	4	1
8	MBConv6, k3x3	320	1	1
9	Conv1x1 & Pooling & FC	1,280	-	1

Table D.2: EfficientNet-Lite B0 - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	MBConv1, k3x3	16	1	1
3	MBConv6, k3x3	24	2	2
4	MBConv6, k5x5	40	2	2
5	MBConv6, k3x3	80	3	2
6	MBConv6, k5x5	112	3	1
7	MBConv6, k5x5	192	4	1
8	MBConv6, k3x3	320	1	1
9	Conv1x1 & Pooling & FC	1,280	-	1

Table D.3: EfficientNet V2-S - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	Fused-MBConv1, k3x3	24	2	1
3	Fused-MBConv4, k3x3	48	4	2
4	Fused-MBConv4, k3x3	64	4	2
5	MBConv4, k3x3	128	6	2
6	MBConv6, k3x3	160	9	1
7	MBConv6, k3x3	256	15	2
8	Conv1x1 & Pooling & FC	1,280	-	1

Table D.4: EfficientNet V2-M - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	Fused-MBConv1, k3x3	24	3	1
3	Fused-MBConv4, k3x3	48	5	2
4	Fused-MBConv4, k3x3	80	5	2
5	MBConv4, k3x3	160	7	2
6	MBConv6, k3x3	176	14	1
7	MBConv6, k3x3	304	18	2
8	MBConv6, k3x3	512	5	1
9	Conv1x1 & Pooling & FC	1,280	-	1

Table D.5: EfficientNet V2-L - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	Fused-MBConv1, k3x3	32	4	1
3	Fused-MBConv4, k3x3	64	7	2
4	Fused-MBConv4, k3x3	96	7	2
5	MBConv4, k3x3	192	10	2
6	MBConv6, k3x3	224	19	1
7	MBConv6, k3x3	384	25	2
8	MBConv6, k3x3	640	7	1
9	Conv1x1 & Pooling & FC	1,280	-	1

Table D.6: MobileNet V2 - places where the FPNs and heads are connected to feature extractor.

Stage	Operator	#Channels	#Layers	#Stride
1	Conv3x3	32	1	2
2	bottleneck1, k3x3	16	1	1
3	bottleneck6, k3x3	24	2	2
4	bottleneck6, k3x3	32	3	2
5	bottleneck6, k3x3	64	4	2
6	bottleneck6, k3x3	96	3	1
7	bottleneck6, k3x3	160	3	2
8	bottleneck6, k3x3	320	3	1
9	bottleneck6, k3x3	1,280	1	1
10	Conv1x1 & Pooling & FC	1,280	-	1

Appendix E

Face recognition

E.1 Accuracy of the ArcFace and QMag models

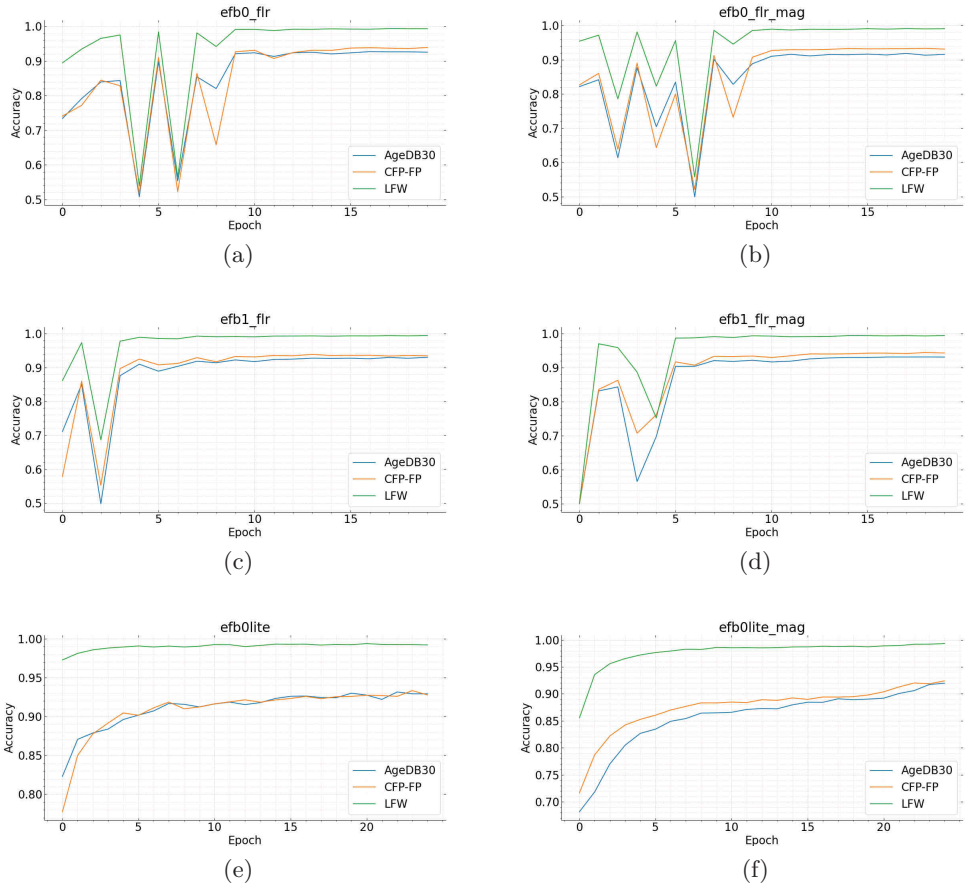


Figure E.1: Accuracy of MagFace and ArcFace models.

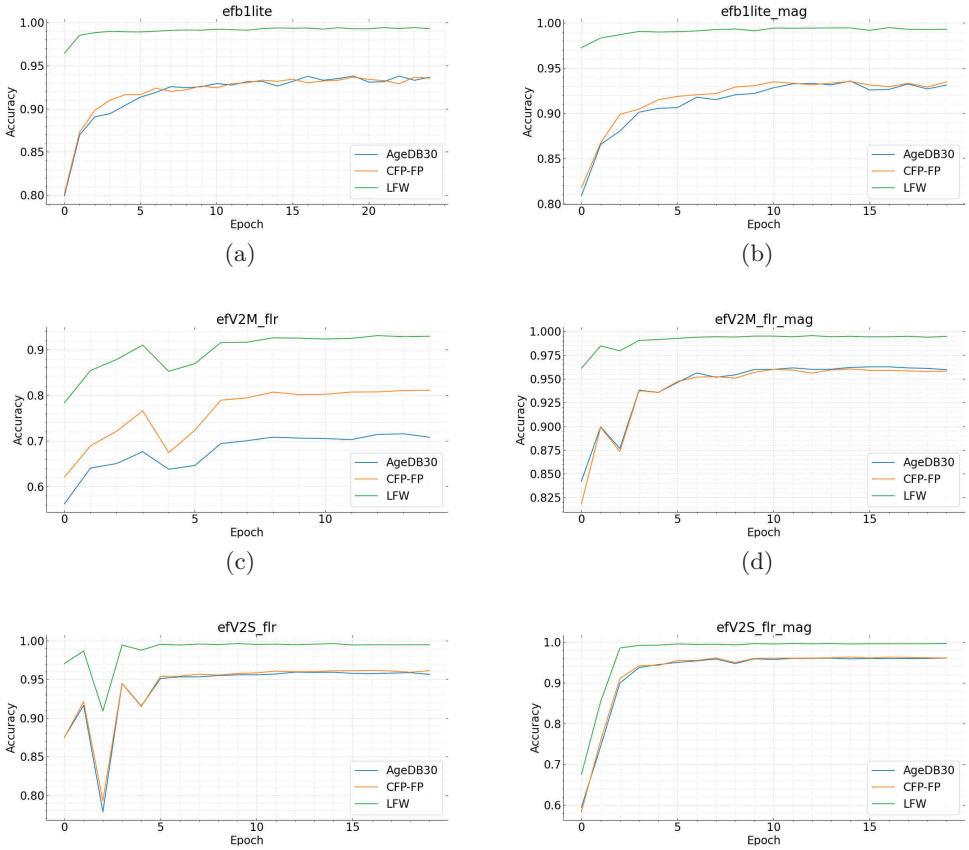


Figure E.2: Accuracy of MagFace and ArcFace models.

E.2 ROC of face recognition models on LFW and AgeDB30

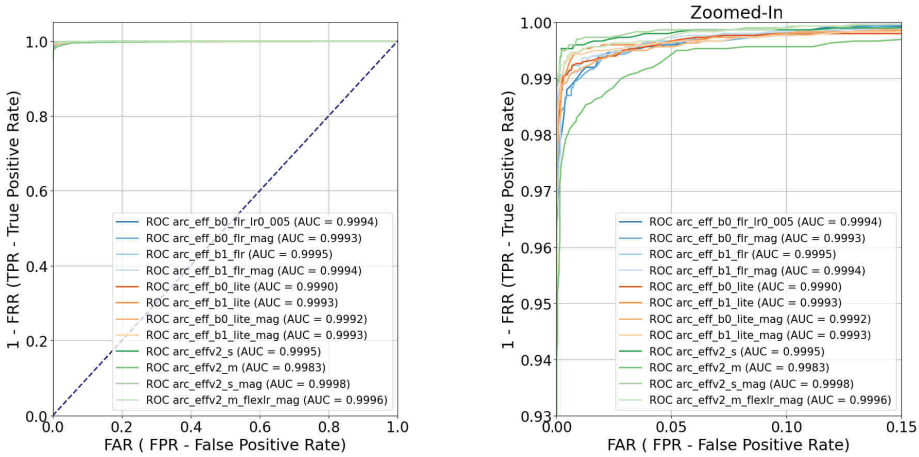


Figure E.3: Evaluation on LFW dataset - ROC curves.

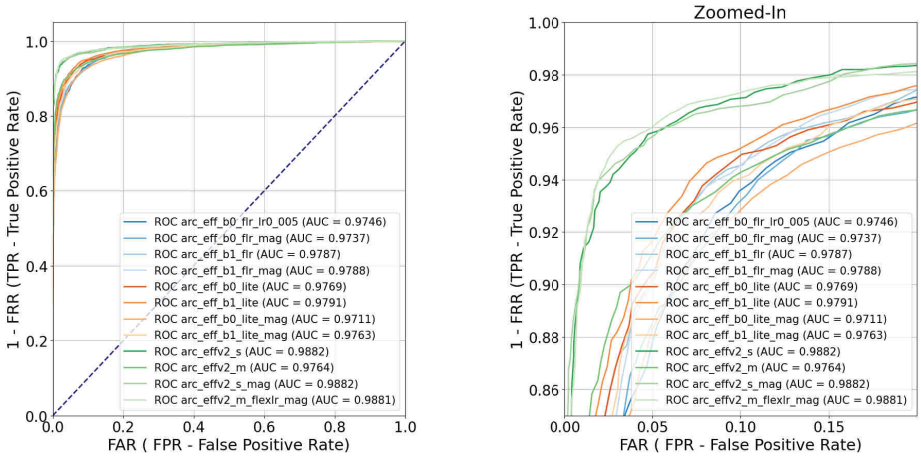


Figure E.4: Evaluation on AgeDB30 dataset - ROC curves.

Appendix F

Face image preprocessing

F.1 Non-Local Means Denoising

The Non-Local Means Denoising [239] is based on the following equations

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q) w(p,q), \quad C(p) = \sum_{q \in B(p,r)} w(p,q), \tag{F.1}$$

where u_i is the i^{th} channel of the image $u = (u_1, u_2, u_3)$ and p expresses a certain pixel in the image. $B(p,r)$ is a function indicating a neighborhood centered at p with size $(2r+1)\times(2r+1)$. The weight $w(p,q)$ is based on the Euclidean distance $d^2 = d^2(B(p,f), B(p,f))$ of the $(2f+1)\times(2f+1)$ then

$$d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{j \in B(0,f)} \left(u_i(p+j) - u_i(q+j)\right)^2. \tag{F.2}$$

The weights function $w(p,q)$ is defined as

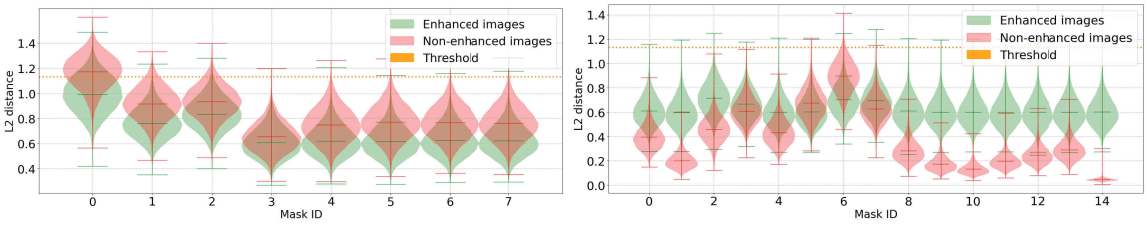
$$w(p,q) = e^{-\frac{\max\left(d^2 - 2\sigma^2, 0.0\right)}{h^2}}, \tag{F.3}$$

where h is the filtering parameter and σ denotes the standard deviation of the noise. The weighting function is set in such a way that similar patches are averaged up to the noise level.

This is the basic principle of the algorithm, but there are several other variations designed to further improve both Peak Signal to Noise Ratio (PSNR) and computational speed.

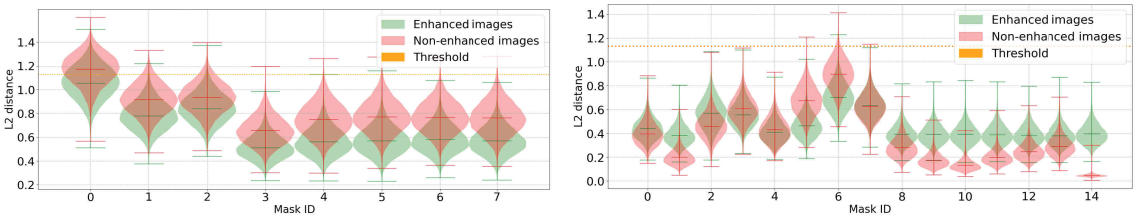
F.2 Experiments

The following figures show the distance distributions for other kernel sizes.



(a) Hard shadows (b) Soft shadows

Figure F.1: Distributions before and after enhancement for kernel 3×3 without denoising.



(a) Hard shadows (b) Soft shadows

Figure F.2: Distributions before and after enhancement for kernel 15×15 without denoising.