University of Hertfordshire
School of Engineering and Technology

DESIGN AND REAL-WORLD APPLICATION OF NOVEL MACHINE LEARNING TECHNIQUES FOR IMPROVING FACE RECOGNITION ALGORITHMS

Daniel Sáez Trigueros

Submitted to the University of Hertfordshire in partial fulfilment
of the requirements for the degree of Doctor of Philosophy
November 2018

Abstract

Recent progress in machine learning has made possible the development of real-world face recognition applications that can match face images as good as or better than humans. However, several challenges remain unsolved. In this PhD thesis, some of these challenges are studied and novel machine learning techniques to improve the performance of real-world face recognition applications are proposed.

Current face recognition algorithms based on deep learning techniques are able to achieve outstanding accuracy when dealing with face images taken in unconstrained environments. However, training these algorithms is often costly due to the very large datasets and the high computational resources needed. On the other hand, traditional methods for face recognition are better suited when these requirements cannot be satisfied. This PhD thesis presents new techniques for both traditional and deep learning methods. In particular, a novel traditional face recognition method that combines texture and shape features together with subspace representation techniques is first presented. The proposed method is lightweight and can be trained quickly with small datasets. This method is used for matching face images scanned from identity documents against face images stored in the biometric chip of such documents. Next, two new techniques to increase the performance of face recognition methods based on convolutional neural networks are presented. Specifically, a novel training strategy that increases face recognition accuracy when dealing with face images presenting occlusions, and a new loss function that improves the performance of the triplet loss function are proposed. Finally, the problem of collecting large face datasets is considered, and a novel method based on generative adversarial networks to synthesize both face images of existing subjects in a dataset and face images of new subjects is proposed. The accuracy of existing face recognition algorithms can be increased by training with datasets augmented with the synthetic face images generated by the proposed method. In addition to the main contributions, this thesis provides a comprehensive literature review of face recognition methods and their evolution over the years.

A significant amount of the work presented in this PhD thesis is the outcome of a 3-year-long research project partially funded by Innovate UK as part of a Knowledge Transfer Partnership between University of Hertfordshire and IDscan Biometrics Ltd[1] (parthership number: 009547).

[1]Now part of GBG plc as GBG IDscan.

Acknowledgments

I would like to thank my supervisors Dr Lily Meng and Dr Margaret Hartnett for their invaluable guidance and support. Contrasting their different points of view has greatly stimulated my thinking and improved the quality of my research. I would also like to thank Dr Heinz Hertlein, who supervised the first year of my PhD, for his helpful guidance during those first steps.

I feel very fortunate to have had the opportunity to do my research in collaboration with IDscan Biometrics. I would not be writing this today if they had not selected me to work on the KTP project that led to this PhD thesis. I would like to thank Innovate UK for the funding that made it possible, and the fantastic KTP team for their input and feedback on my work. I would especially like to thank my KTP Adviser Gerry O'Hagan for all of his good advice on my personal and professional development. I am also very grateful to all my colleagues at IDscan for making it such a fun and inspiring place to work. I would like to thank my colleagues in the research team for creating such a positive and unique atmosphere for innovation. A special mention to Moneer Alitto for all the interesting discussions and sharing of ideas.

I would also like to thank the good friends that I have made in London in these past four years for making me feel like at home. Finally, I would like to thank my family, especially my parents. No matter how far I go, I can always count on their love, support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Face recognition refers to the technology capable of identifying or verifying the identity of subjects in images or videos. The first face recognition algorithms were developed in the early seventies [1, 2]. Since then, their accuracy has improved to the point that nowadays face recognition is often preferred over other biometric modalities that have traditionally been considered more robust, such as fingerprint or iris recognition [3]. One of the differential factors that make face recognition more appealing than other biometric modalities is its non-intrusive nature. For example, fingerprint recognition requires users to place a finger in a sensor, iris recognition requires users to get significantly close to a camera, and speaker recognition requires users to speak out loud. In contrast, modern face recognition systems only require users to be within the field of view of a camera (provided that they are within a reasonable distance from the camera). This makes face recognition the most user friendly biometric modality. It also means that the range of potential applications of face recognition is wider, as it can be deployed in environments where the users are not expected to cooperate with the system, such as in surveillance systems. Other common applications of face recognition include access control, fraud detection, identity verification and social media.

Face recognition is one of the most challenging biometric modalities when deployed in unconstrained environments due to the high variability that face images present in the real world (these type of face images are commonly referred to as faces *in-the-wild*). Figure 1.1 shows some of these variations, including head poses, aging, occlusions, illumination conditions, and facial expressions. Examples of face recognition methods that are deployed in unconstrained environments are surveillance systems in which users are not aware that they are being recorded, or face verification systems that compare face images captured under any conditions. To simplify the problem, many face recognition algorithms constrain the distribution of face images (sometimes called the *face space*) to be analysed by one or several factors. For example, face recognition systems for border control typically

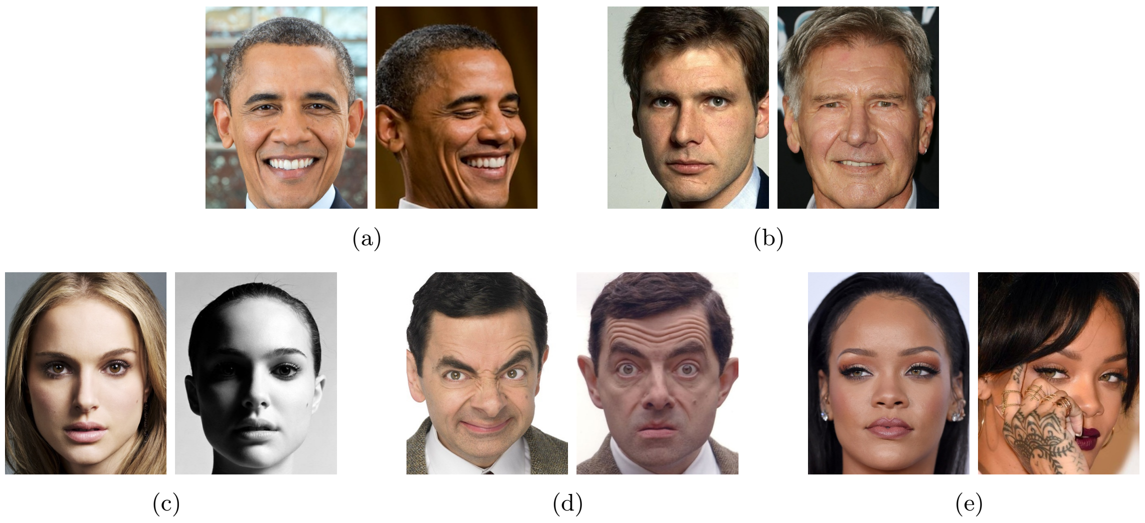


Figure 1.1: Typical variations found in faces in-the-wild. (a) Head pose. (b) Age. (c) Illumination. (d) Facial expression. (e) Occlusion.

constrain the face recognition problem by making the user stand in a well-lit area and look at a high-resolution camera in a frontal pose. This normalisation of the image capturing process reduces the variability that face images present when analysed by a face recognition algorithm. Constraining the face space can be done in many other ways, e.g. by limiting the population size or reducing the maximum allowed age gap when comparing two faces. By taking into account these constraints, face recognition algorithms can be more specialised and achieve a higher accuracy. Note, however, that the distinction between constrained and unconstrained face recognition is not directly related to the applicability of face recognition in the real-world. A face recognition algorithm that works well in unconstrained conditions might not be the best choice when conditions are constrained. For this reason, the aim of this thesis is to propose novel machine learning techniques for improving the performance of face recognition applications that fall in both of these two categories.

Face recognition techniques have shifted significantly over the years. Traditional methods relied on hand-crafted features, such as edges and texture descriptors, combined with machine learning techniques, such as principal component analysis, linear discriminant analysis or support vector machines. The difficulty of engineering features that were robust to the different variations encountered in unconstrained environments made researchers focus on specialised methods for each type of variation, e.g. age-invariant methods [4, 5], pose-invariant methods [6], illumination-invariant methods [7, 8], etc. Recently, traditional face recognition methods have been superseded by deep learning methods based on convolutional neural networks (CNNs). The main advantage of deep learning methods is that they can be trained with very large datasets to learn the best features to represent the data. The availability of faces in-the-wild on the web has allowed the collection of large-scale datasets of faces [9, 10, 11, 12, 13, 14, 15] containing real-world variations. CNN-based face recognition methods trained with these datasets have achieved very high accuracy as they are able to learn features that are robust to the real-world variations present in the face images used during training. Moreover, the rise in popularity of deep learning methods for computer vision has accelerated face recognition research, as CNNs are being used to solve many other computer vision tasks, such as object detection and recognition, segmentation, optical character recognition, facial expression analysis, age estimation, etc.

Although significant progress has been made in recent years, a number of challenges remain. On one hand, methods based on CNNs have proven to be very effective for many applications. However, these type of methods require a large amount of training images to work well and can be very computationally expensive. As will be shown in Chapter 4, traditional methods are still relevant as they can achieve good accuracy in constrained settings. On the other hand, even when enough face images and computational resources are available to train CNN methods, the problem of recognising faces in-the-wild remains unsolved. This is because it is difficult to collect datasets that contain all the variations that will be seen in a real-world deployment. For this reason, reducing the data requirements of CNN methods is a fundamental problem. Chapters 5 and 6 of this thesis propose new techniques to tackle this issue, both by using better training schemes and by augmenting face datasets with synthetic data.

1.2 Outline

The body of this thesis is divided as follows:

- **Chapter 2** provides the background needed for understanding traditional and deep learning approaches for computer vision applications.
- **Chapter 3** provides an introduction to face recognition and a comprehensive literature review of popular face recognition methods.
- **Chapter 4** focuses on a particular constrained face recognition problem and proposes a new hybrid face recognition method to tackle it.
- **Chapter 5** proposes two new techniques to improve the performance of CNN-based methods for the recognition of faces in-the-wild.
- **Chapter 6** proposes a novel generative model to augment databases of faces with synthetic face images that can be used for training.
- **Chapter 7** provides this thesis' general conclusions by recapitulating the contributions and proposing further work for the future.

1.3 Contributions

This thesis' contributions to the field are presented in Chapters 4 to 6. Although these chapters are not directly related, they follow a common theme of improving face recognition in real-world applications. They are written in a publication-style format as they are adaptations of academic papers (at the time of writing, a version of Chapter 4 [16] has been published in a conference proceedings; a version of Chapter 5 [17] has been published as a journal article; and a version of Chapter 6 is being prepared for submission). This section gives a short summary of each of these chapters and highlights the contributions made in this thesis.

Despite the recent success of deep learning methods, some face recognition applications are simple enough that can be solved with less computationally expensive methods that do not need to be trained with very large datasets. In Chapter 4 a traditional face recognition method to solve such a task is presented. In particular, the proposed method is used to match a face image scanned from an identity document presenting security features (e.g. watermarks and holograms) against a face image stored in the biometric chip of such a document. The proposed method uses a novel combination of texture and shape features together with subspace representation techniques. In addition, the adoption of two operating points to enhance the reliability of the final verification decision is proposed.

In Chapter 5 two contributions that improve the performance of CNNs for face recognition are presented. First, it is shown how facial occlusions remain a challenge when training a CNN using a dataset of faces collected from the web containing a low amount of face images presenting occlusions. Then, a technique to identify in which face regions occlusions can affect face recognition accuracy the most, and a data augmentation method that forces the CNN to learn discriminative features from all the face regions equally are proposed. The second contribution in this chapter is a modification of the popular triplet loss function often used when training CNNs for face recognition. More specifically, the proposed loss function can achieve a greater separability between the distributions of positive and negative scores by not only separating their means but also minimising their standard deviations.

Training data is arguably the most important component needed to achieve high accuracy using deep learning methods. However, collecting large-scale, good quality datasets is not always an easy task. In Chapter 6 this issue is addressed by proposing a novel generative adversarial network that can be used to synthesize both face images of existing subjects in a dataset and face images of new subjects. The experiments in this chapter show how models trained with different combinations of real data and synthetic data generated by the proposed generative model achieve better recognition accuracy than models trained with real data alone.

Chapter 2

Background

Computer vision is a field of computer science that aims to give computers the means to understand visual information in the same way as humans. All computer vision tasks require a high-level understanding of the numeric values that computers use to represent images. Some examples of classic computer vision applications include object recognition, scene understanding, optical character recognition, image restoration and face recognition.

This chapter provides some technical background on the computer vision techniques that are most relevant to face recognition in general and to the understanding of this thesis in particular. The chapter is divided into three sections, starting from a basic approach of using hand-engineered features to extract information from images to more scalable and automatic approaches of using machine learning and deep learning for feature selection and feature learning.

2.1 Feature Engineering

In a computer system, images are represented by pixels arranged in a 2-dimensional grid. These pixels comprise numeric values that indicate colour intensity at each spatial location in an image. In the case of colour images, each pixel consists of multiple values, one for each colour channel (typically red, green and blue). One of the main challenges when dealing with images comes from the large number of possible combinations of pixel values that can yield an image. For example, the number of possible combinations of pixels in a 8-bit greyscale image with a low resolution of 32×32 pixels is $2^{8\times32\times32}$. Even for such a low-resolution image, this number is extremely large and one of the reasons why inferring meaningful information about the content of an image solely by looking at raw pixel values is not possible in general.

From a pattern recognition perspective, images are said to suffer from the curse of dimensionality as a result of their high-dimensionality. More specifically, when dealing with high-dimensional data such as images, the use of traditional statistical analysis methods to find patterns in the data becomes unfeasible since the number of examples available to solve a problem is typically not enough to cover all the possible modes of variation. To overcome this problem, researchers have developed dimensionality reduction methods that extract relevant information from the data in the form of *features*. At its core, feature engineering refers to the process of using domain-specific knowledge to develop features that transform high-dimensional data representations into discriminative lower-dimensional data representations.

2.1.1 Shape Features

One of the basic features used in computer vision is the presence of edges in an image. These edges correspond to sharp transitions in pixel intensity, or equivalently, to high-frequency contents in an image. The most common approach to computing edges is to perform high-pass filtering on an image by approximating the image gradient at each spatial location therein. Since the gradient is the directional derivative of the intensity in an image, locations that yield large gradients display large changes in pixel intensity, whereas locations that yield small gradients display constant pixel intensity.

An approximate image gradient is typically computed in the spatial domain by convolving an image with a small filter (also called kernel). For example, given an image $\boldsymbol{I}$, the horizontal and vertical gradient images $\boldsymbol{G}_x$ and $\boldsymbol{G}_y$ can be computed using the well-known Sobel operator [18]:

$$\boldsymbol{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \boldsymbol{I} \tag{2.1}$$

$$\boldsymbol{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \boldsymbol{I} \tag{2.2}$$

The spatial convolution operation (denoted by $*$) involves sliding the filter across all spatial locations in an image and computing the dot product between the corresponding pixels and the values in the filter. The final gradient image $\boldsymbol{G}_{xy}$ can be computed as the gradient magnitude of both directional components:

$$\boldsymbol{G}_{xy} = \sqrt{\boldsymbol{G}_x^2 + \boldsymbol{G}_y^2} \tag{2.3}$$

Figure 2.1 illustrates how the Sobel operator can be used to effectively detect edges in an image.

Many early computer vision methods used edge detection as a mechanism for simplifying images. For example, Larry Robert's Blocks World [19] proposed a way of inferring the 3D shape of objects by extracting and grouping their edges. Subsequently, some approaches were proposed to recognise more complex structures by combining basic object shapes [20, 21].

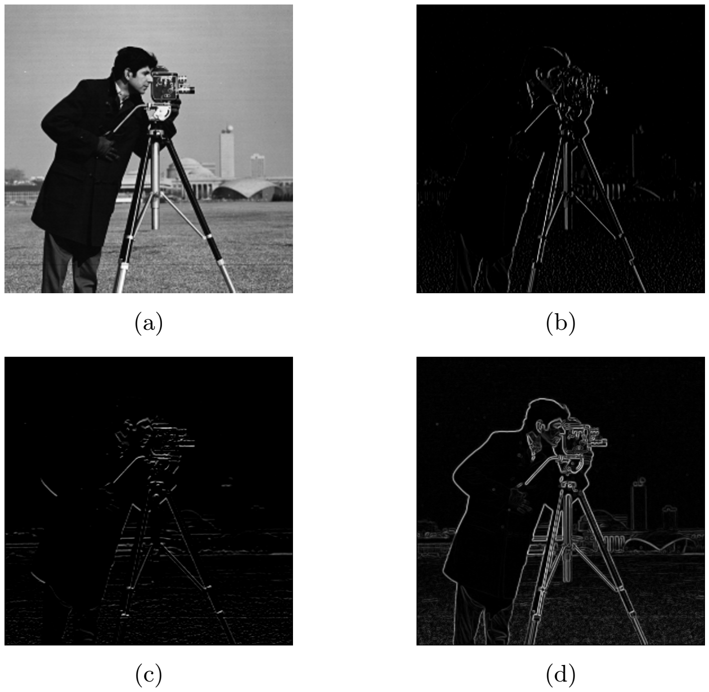


Figure 2.1: Images filtered using the Sobel operator (a) Original image. (b) Horizontal gradient image $\boldsymbol{G}_x$. (c) Vertical gradient image $\boldsymbol{G}_y$. (d) Gradient image $\boldsymbol{G}_{xy}$.

More advanced methods such as active contours models [22] and active shape models [23] are able to outline complex object shapes using edge information. In particular, active shape models use a deformable model of the assumed shape of an object, namely a *point distribution model*, that is iteratively fitted to new examples. The basic approach is as follows:

1. Create a point distribution model (PDM) using a training set of examples with annotated landmarks.

2. Initialise the contour of a new object using the PDM.

3. Look for nearby edges in directions perpendicular to the current contour to generate a new contour that is closer to the edges of the object. The new contour is constrained by the PDM.

4. Repeat step 3 until convergence is achieved.

Active shape models and its variations are widely used to track the shape of deformable objects such as body parts. Figure 2.2 shows an application of active shape models for facial landmark location.

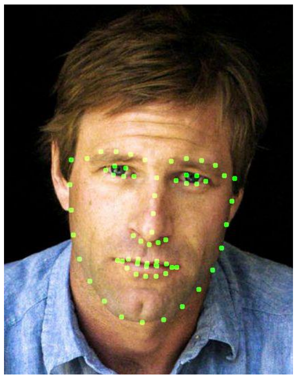


Figure 2.2: Facial landmark location using active shape model.

2.1.2 Texture Features

Although edges and shapes are very useful features for describing images, many computer vision tasks require more sophisticated solutions. For example, predicting whether two face images belong to the same subject based on the shape and locations of the facial landmarks alone would not provide very accurate results. This is because many faces have very similar shapes, as well as the fact that the face shape of a subject varies significantly due to different factors such as facial expression and head pose. In this case, it is also useful to have a representation of the texture of the face. This can be done by analysing the texture of the image at different *keypoints* using texture descriptors.

For example, one popular texture descriptor is the *local binary pattern* (LBP) [24]. The basic method for computing LBP descriptors in an image is as follows:

1. Define a neighbourhood of fixed size around each pixel, for example, a 3×3 window.
2. Compare the centre pixel of each neighbourhood to each of its neighbours, giving a value of 0 if the centre pixel is greater than the neighbour and 1 otherwise. This gives a binary number (8 binary digits in the case of a 3×3 window) at each location which can be converted to a decimal number for convenience. This process is illustrated in Figure 2.3.
3. Create a histogram containing the frequency of each pattern (for a 3×3 window, the histogram has 2^8 bins). This histogram is the final feature vector and can be computed over the whole image or over smaller regions.

LBP descriptors are fast to compute and provide a good estimation of the global texture of an image. Many variants of LBP have been proposed [25] and used to solve a wide range of applications such as face recognition [26], age classification [27] or palm print identification [28].

Considerable research has been done on feature extractors that are invariant to scale, orientation, illumination and other variations. These feature extractors typically comprise two stages, namely, feature localisation and feature description. During the feature localisation stage, a number of keypoints are automatically selected in the image. The goal is to select keypoints at locations in the image that remain detectable across different instances of the same object. At the same time, the keypoints need to be located in descriptive regions of the image (e.g. regions displaying constant pixel intensity should be avoided). In the feature description stage, a discriminative texture descriptor is computed around each selected keypoint. These descriptors can be used to match features between objects. Figure 2.4. shows an example of feature matching using the *scale-invariant feature transform* (SIFT) [29], a popular feature extractor that follows this approach. Other similar methods include *speeded up robust features* (SURF) [30] and *binary robust invariant scalable keypoints* (BRISK) [31].

Another approach to feature extraction is *dense* feature extraction. In this case, instead of automatically selecting keypoints, the texture descriptors are computed on a grid of equidistant points. This is particularly useful when working with aligned images since the grid forces features to be extracted exactly at the same locations in every image. Figure 2.5 shows the difference between keypoint feature extraction and dense feature extraction on two aligned faces. Observe how, in this case, using dense features (Figures 2.5c and 2.5d) provides a much more accurate location for extracting consistent features across both images.

2.1.3 Other Features

Shape and texture features are the most commonly used features in computer vision as they can be computed on any image. More specialised features can be useful in certain situations. For example, features that use colour information can be useful for some problems such as skin colour detection [32], and motion features such as optical flow [33] are often used when dealing with videos. For a comprehensive and up-to-date survey of many of the features often used in computer vision, see [34].

2.1.4 Features in Practice

This section has shown how to obtain more descriptive and lower-dimensional representations of images by extracting different types of features. While features alone can be used to solve some simple problems, in general, they are not sufficient to solve more complex problems. For example, consider a face recognition system that relies on densely matching SIFT descriptors as shown in Figures 2.5c and 2.5d. Even if a grid of only 10×10 points is used, the resulting representation would still have a

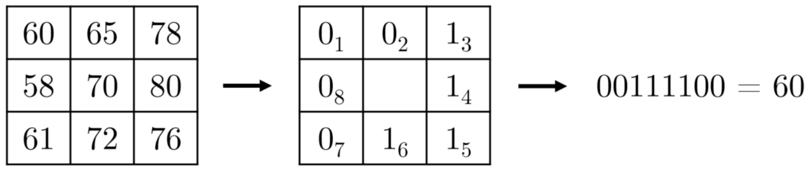


Figure 2.3: Computation of an LBP descriptor over a 3×3 window.

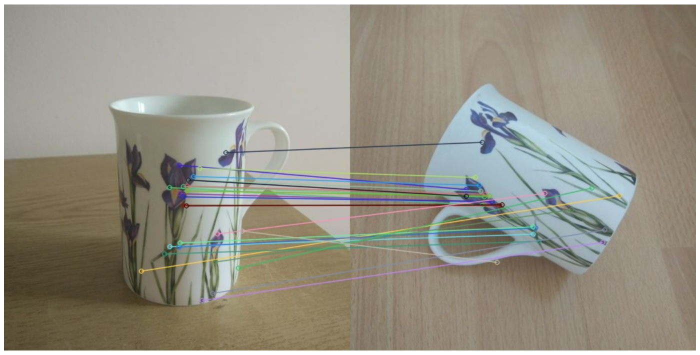


Figure 2.4: Feature matching using SIFT.

very high-dimensionality ($10 \times 10 \times 128$, since each SIFT descriptor has 128 dimensions). Thus, it is still likely to suffer from the curse of dimensionality. This problem can be solved by dimensionality reduction techniques that use statistical analysis to find the most important dimensions, such as *principal component analysis* (PCA). Moreover, a lot of problems cannot be solved by just extracting and matching features. These include classification problems which involve predicting the category to which an example belongs (e.g. optical character recognition), and regression problems which involve predicting a continuous value associated with a particular attribute of the data (e.g. age estimation).

Dimensionality reduction, classification and regression are typical examples of tasks that can be solved using machine learning techniques. In practice, most computer vision problems are solved by combining features with machine learning techniques. The next section describes basic machine learning concepts and some of the common algorithms used in computer vision applications.

2.2 Machine Learning

Machine learning is a field of artificial intelligence concerned with the development of algorithms that can learn from data. Machine learning algorithms can learn to perform tasks without being explicitly programmed to do so. This is useful in any situation where defining such tasks using a traditional computer programming paradigm would be overly difficult. Machine learning algorithms can be broadly classified as supervised and unsupervised learning techniques depending on how they use data to learn.

2.2.1 Supervised Learning

Supervised learning algorithms involve learning a function f that maps inputs $\boldsymbol{X}$ to outputs $\boldsymbol{Y}$, i.e. $f : \boldsymbol{X} \to \boldsymbol{Y}$. To learn this mapping, a supervised learning algorithm must be trained with a set of n labelled examples $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ where $\boldsymbol{x}_i$ represent an input example and $\boldsymbol{y}_i$ the desired

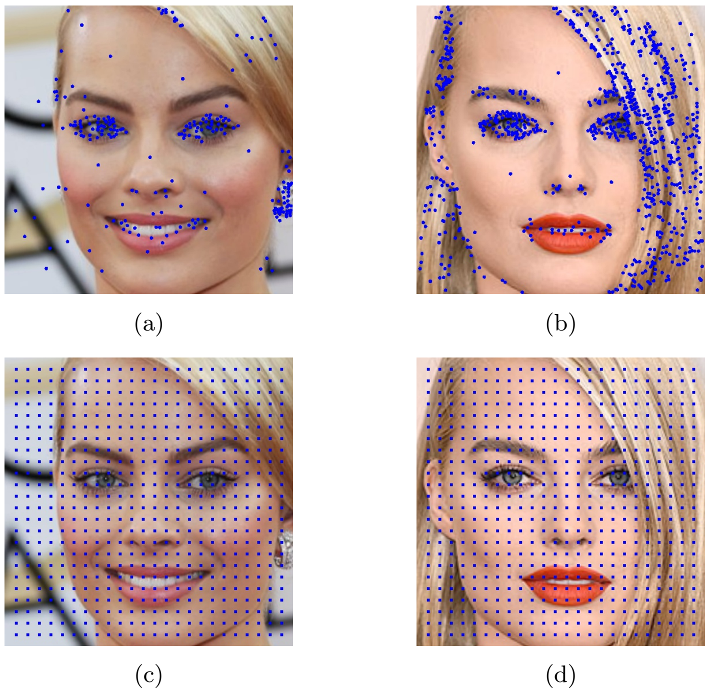


Figure 2.5: Keypoint vs dense feature extraction on aligned images. (a) and (b) SIFT keypoints. (c) and (d) Grid of equidistant points.

output for that particular example. In computer vision, the inputs are typically pixels or features and the outputs are what needs to be predicted, e.g. coordinates of points in an image, probability that an object is present in an image, etc.

Since the goal is to develop algorithms that generalise to examples outside the training set, it is assumed that all the available examples for training and testing are independent and identically distributed (i.i.d.) samples from a common data generating distribution. The i.i.d. assumption implies that a function f that provides a good fit on training data should also provide a good fit on test data. A discussion of what a *good fit* actually means is postponed to Section 2.2.5.

To find the function that best approximates the desired mapping, a loss or objective function needs to be defined. A loss function measures the error between a desired output $\boldsymbol{y}$ and a predicted output $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$. In general, most machine learning models learn functions that are controlled by a set of parameters $\boldsymbol{\theta}$. Therefore, the goal is to find the optimal parameters $\boldsymbol{\theta}^*$ from the set of all possible values $\boldsymbol{\Theta}$ that minimise the loss over all the training examples:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min} \sum_{i}^{n} L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i; \boldsymbol{\theta}) \tag{2.4}$$

As a practical example, let us consider a simple binary linear classifier trained with a dataset of n 2-dimensional examples, each associated with a binary label, i.e $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n) \mid \boldsymbol{x}_i \in \mathbb{R}^2, y_i \in \{0,1\}\}$. Since the model is linear, the algorithm needs to find a linear function of the inputs $w_1x_1 + w_2x_2 + b$ that separates the two classes, as illustrated in Figure 2.6. The output of a binary linear classifier is as follows:

$$\hat{y} = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) \tag{2.5}$$

where

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.6}$$

is a sigmoid function that squashes the output of the linear mapping to a value between 0 and 1; and $\boldsymbol{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ are the parameters of the model. The output of the classifier $\hat{y}$ can be interpreted as the probability of the predicted class being 1, whereas its complement $1 - \hat{y}$ can be interpreted as the probability of the predicted class being 0. Next, a loss function needs to be defined. In classification problems, it is common to use the cross-entropy loss, which for the binary case takes the form:

$$L(\hat{y}, y) = -y \log \hat{y} - (1-y)\log(1-\hat{y}) \tag{2.7}$$

It can be shown that minimising the cross-entropy loss is equivalent to minimising the Kullback-Leibler divergence (which is a measure of distance between probability distributions) between the predicted probability distribution and the true probability distribution. The final step is to use an optimisation procedure to find the parameters $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ that minimise the average loss over the training set (Equation 2.4). Once the classifier is trained, its parameters are fixed and Equation 2.5 is used to predict the class of new samples.

The binary linear classifier can be easily generalised to a multiclass linear classifier. In this case, each example is associated with a class label $\boldsymbol{y}$ that is a k-dimensional vector containing 0 at all indices except a 1 at the index of the correct class, where k is the number of classes in the dataset. For example, for a classification problem with 5 classes, a sample belonging to the second class would have a label $\boldsymbol{y} = [0,1,0,0,0]$. The output of the linear mapping in a multiclass linear classifier is a vector of k class scores that is normalised to a vector of k class probabilities that sum to one using the softmax function. Thus, the output of a multiclass linear classifier is:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) \tag{2.8}$$

where $\boldsymbol{W} \in \mathbb{R}^{2\times k}$, $b \in \mathbb{R}^k$ and

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_j^k e^{z_j}} \tag{2.9}$$

For a multiclass classifier, the general form of the cross-entropy loss is used:

$$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_j^k y_j \log \hat{y}_j \tag{2.10}$$

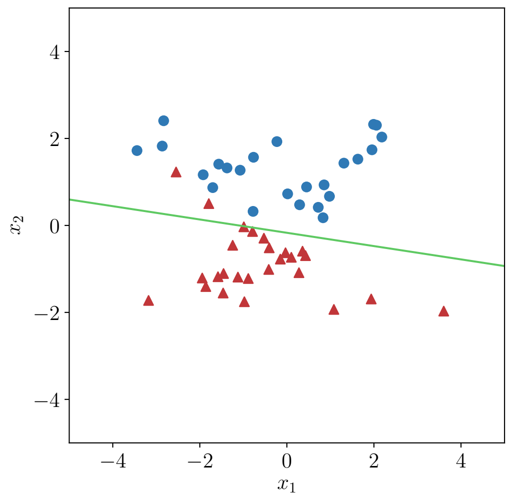


Figure 2.6: Linear classifier decision boundary.

More complex non-linear classifiers such as neural networks, discussed in Section 2.3, are very similar to the linear classifiers presented above but use non-linear functions to map inputs to a linearly separable space.

2.2.2 Unsupervised Learning

Unsupervised learning algorithms involve learning when the target outputs $\boldsymbol{Y}$ are not available. Unsupervised learning tasks typically require learning something about the data generating distribution. For example, clustering is an unsupervised learning task which consists of dividing a dataset into groups of examples with similar properties (e.g. grouping images that contain similar objects).

A family of unsupervised learning algorithms relevant to this thesis are unsupervised generative models. In these models, the goal is to learn the data generating distribution itself. This can be useful for generating samples that are similar to those in the training set, by sampling from the estimated probability distribution learnt by the model. Generative models have gained a lot of attention recently in the deep learning community. Hence, their discussion is deferred to Section 2.3.

As a practical example of an unsupervised learning algorithm, let us consider principal component analysis (PCA). PCA is a technique that uses an orthogonal linear transformation to project input data onto a space of uncorrelated dimensions. These dimensions are referred to as the principal components and they define the directions along which the variance in the data is the highest. An illustration of PCA is shown in Figure 2.7. PCA can be used for dimensionality reduction and data compression by selecting a subset of the principal components that account for the most variance as

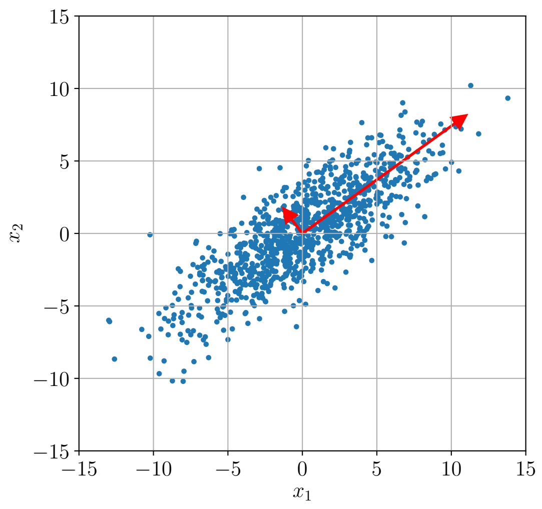


Figure 2.7: Principal components of a 2D multivariate Gaussian distribution. The length of each arrow is proportional to the amount of variance in each direction.

the new basis vectors. The PCA dimensionality reduction algorithm can be summarised as follows:

1. Given a dataset of n d-dimensional examples $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ in its matrix form $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, normalise each feature to have zero mean and unit variance. This transformation ensures that the data is centred and that all the features have the same range of values.

2. Find the eigenvectors and eigenvalues of the covariance matrix $\text{cov}(\boldsymbol{X}) = \frac{1}{n}\sum_i^n \boldsymbol{x}_i \boldsymbol{x}_i^T$. The eigenvectors of the covariance matrix are the principal components and the eigenvalues indicate the amount of variance that each principal component accounts for.

3. Select the $k < d$ eigenvectors that correspond to the k largest eigenvalues. This can be done by fixing the output dimensionality to k principal components or by fixing the amount of variation to be retained and calculating how many k principal components are needed.

4. Construct a matrix $\boldsymbol{W} \in \mathbb{R}^{d \times k}$ of basis vectors from the k selected principal components.

5. Project the original dataset $\boldsymbol{X}$ onto the space spanned by the k principal components to obtain a lower-dimensional representation $\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W}$ where $\boldsymbol{T} \in \mathbb{R}^{n \times k}$.

The derivation of this algorithm is beyond the scope of this chapter and can be found in machine learning textbooks such as [35, 36].

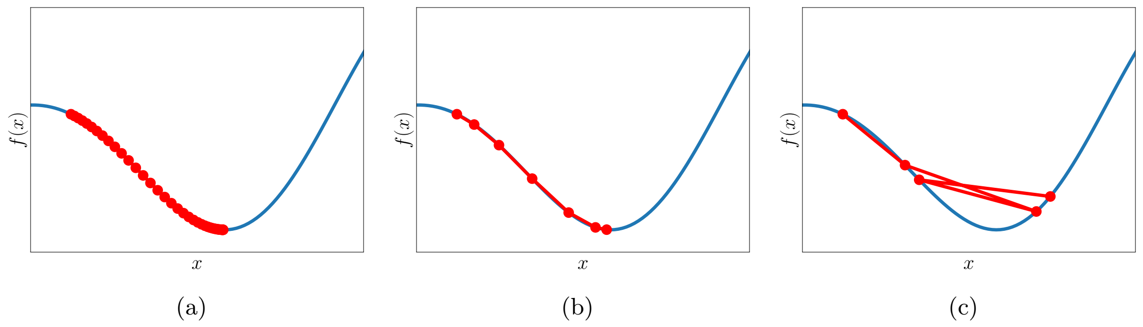


Figure 2.8: Effect of the learning rate ϵ. Each red marker represents a gradient descent step. (a) Small learning rate. (b) Appropriate learning rate. (c). Large learning rate.

2.2.3 Optimisation

Even though the solution of simple algorithms such as PCA can be obtained analytically, most machine learning algorithms require some form of optimisation to find their solutions. By optimisation it is meant a procedure that finds the optimal value of x that minimises (or maximises) a function $f(x)$. This is usually denoted as $x^* = \arg\min f(x)$ (or $\arg\max$ for maximisation). The most common form of optimisation used in machine learning is gradient-based optimisation.

Gradient-based optimisation can be described as follows: given a function $f(x)$, its first derivative $\frac{\partial f(x)}{\partial x}$ indicates how a small change in the input x affects the output $f(x)$. If the first derivative is positive, a small increment to x will result in a small increment to $f(x)$. If the first derivative is negative, a small increment to x will result in a small decrement to $f(x)$. Therefore, to find the value that minimises $f(x)$, small changes to x need to be made with a sign that is opposite to the sign of the derivative. This optimisation procedure is known as *gradient descent*. More generally, when the input $\boldsymbol{x}$ is a vector, its gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ is a vector containing all the partial derivatives $\frac{\partial f(\boldsymbol{x})}{\partial x_i}$. In this case, the value that minimises $f(\boldsymbol{x})$ can be found by making small changes to $\boldsymbol{x}$ in the direction opposite to the gradient:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{2.11}$$

where ϵ is a step size, commonly referred to as *learning rate*, that scales the gradient update. If the learning rate is too small, the optimisation procedure might take a very long time to converge. On the other hand, if the learning rate is too large, the size of the increment to $\boldsymbol{x}$ might overshoot the minimum of $f(\boldsymbol{x})$ and the optimisation might not converge. The effect of the learning rate is illustrated in Figure 2.8.

Machine learning models are typically optimised by finding the parameters $\boldsymbol{\theta}$ that minimise a loss function L over all the training samples, as shown in Equation 2.4. The gradient of the loss can be most accurately estimated by computing the average gradient over all the training samples:

$$g(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{n} \nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i; \boldsymbol{\theta}) \tag{2.12}$$

In practice, the training sets used in machine learning can be very large, so it is common to estimate the gradient using a batch of $m \ll n$ samples. Since computing the gradient over m samples is computationally cheaper than over n samples, gradient descent can perform more frequent updates and hence converge faster. Note that if the batch size is too small, the gradient might be estimated poorly and the optimisation might take longer to converge. This version of gradient descent is known as *stochastic gradient descent* (SGD) [37]. The basic version of SGD is shown in Algorithm 1.

Algorithm 1 Stochastic gradient descent.

```
Require: Learning rate $\epsilon$
Require: Initial parameters $\boldsymbol{\theta}$
  while not converged do
      Sample a batch of $m$ samples from the training set $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_m, \boldsymbol{y}_m)\}$.
      Estimate gradient $g(\boldsymbol{\theta}) \approx \frac{1}{m}\sum_i^m \nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i; \boldsymbol{\theta})$.
      Update parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon g(\boldsymbol{\theta})$
  end while
```

Most SGD implementations modify Algorithm 1 by decreasing the learning rate as the training progresses so that smaller steps are taken near the minimum of the loss function. Many variations of the update rule of the basic SGD algorithm exist. For example, the momentum update [38] can speed up the optimisation procedure and help it escape from shallow local minima by updating in the direction of an exponentially decaying average of past gradients. Other methods such as AdaGrad [39] and Adam [40] are able to automatically adapt the learning rate for each parameter θ_i separately.

2.2.4 Hyperparameters

Most machine learning algorithms are controlled by a set of parameters that are not learnt by the model itself, e.g. the learning rate. These parameters are commonly referred to as *hyperparameters*. The usual way to select hyperparameters is to try different values and select the ones that provide the best results when the model is evaluated on a disjoint subset of the training set called the validation set. The final performance of the model is evaluated on a test set that should not be used to select the model's parameters or hyperparameters. Therefore, the test set also needs to be disjoint from the training and validation sets.

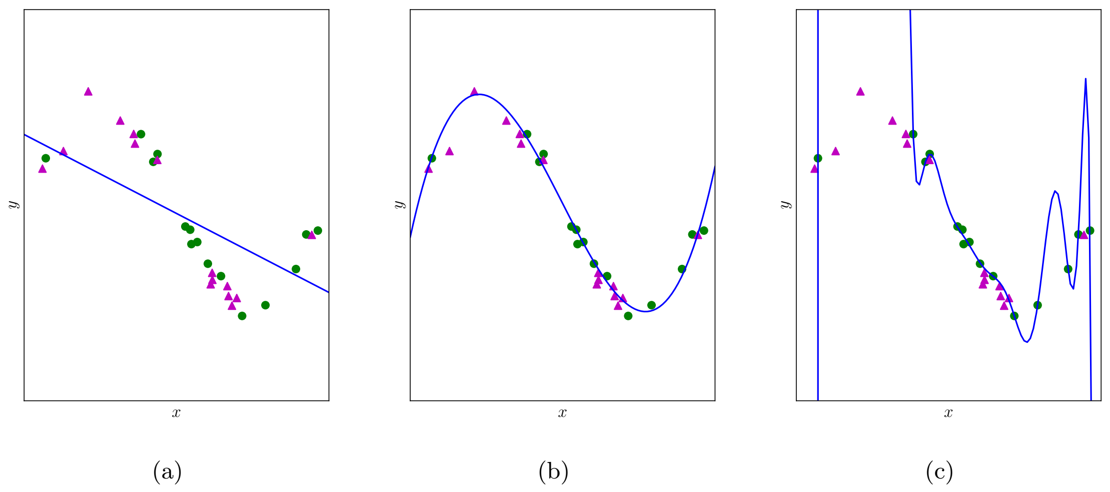


Figure 2.9: Lineal regression model using polynomials of different degrees to control the capacity of the model. The green circles represent points in the training set and the magenta triangles represent points in the test set. (a) **Low capacity model**. A polynomial of degree one underfits as it cannot fit the points in the training set. (b) **Appropriate capacity model**. A polynomial of degree three provides a good fit for the points in the training and test sets. (c) **High capacity model**. A polynomial of degree twelve overfits as it can fit the points in the training set but not the points in the test set.

2.2.5 Model Capacity

An important hyperparameter is the capacity of a model. The representational capacity of a model refers to the model's ability to fit complex functions. For example, the linear classifier presented in Section 2.2.1 has low capacity as it can only represent first-order polynomials, as shown in Equation 2.5. The capacity of that model can be increased by allowing the linear classifier to learn higher degree polynomials of the form $b + \sum_i^k w_i^T x^i$, where k varies the capacity of the model. Models with low capacity can only learn simple functions and might result in the model performing poorly on the training data. This is known as *underfitting*. Models with high capacity can learn overly complicated functions that perform very well on the training data but poorly on the test data. This is known as *overfitting*. Figure 2.9 shows the effect of changing the representational capacity of a simple linear regression model. Notice how in the underfitting case the model is not able to fit the training data, whereas in the overfitting case the model fits the noise in the training data rather than the true data distribution.

Instead of controlling the capacity of a model by restricting the type of functions that the model can learn, a technique known as *regularisation* can be used to give the model a preference for some functions over others. For example, the L_2 regularisation [41] method adds a term to the loss function that controls the capacity of the model by giving a preference for learning small weights. Functions with small weights are smoother (i.e. they have less ups and downs) and therefore are less prone to overfitting.

2.2.6 Machine Learning for Computer Vision

As explained in Section 2.1, feature engineering is an important step in most computer vision applications due to the high-dimensionality of images and videos. Even advanced machine learning algorithms such as support vector machines (SVM) [42] perform better when they are fed with low-dimensional features instead of raw pixel values.

For example, an image classification problem might be solved by densely extracting a large number of features from an image (e.g. LBP descriptors), performing dimensionality reduction (e.g. using PCA) and feeding the resulting lower-dimensional feature vector into a classifier (e.g. SVM). Even though such pipelines work well in many applications, their performance is often limited by the choice of features. Deep learning, discussed in the next section, eliminates the need to hand-engineer features, which are typically suboptimal, by automatically learning the optimal features for the task at hand.

2.3 Deep Learning

The term deep learning is commonly used to refer to deep artificial neural networks. An artificial neural network is a type of machine learning algorithm composed of interconnected nodes, called artificial neurons, arranged in layers. Simply put, each layer takes the outputs of the preceding layer as its inputs, applies a number of non-linear transformations to them, and sends its outputs to the next layer. The term *deep* is used when many layers are stacked together. The main motivation for using deep neural networks instead of other machine learning algorithms is that they are able to learn powerful features from raw data thanks to their hierarchical architecture and non-linear behaviour. This section provides a short summary of the main deep learning techniques used in this thesis. For a complete and up-to-date view of deep learning see [43].

2.3.1 Feedforward Neural Networks

The most basic type of neural network architecture is the *feedforward neural network*, also called *multilayer perceptron* (MLP) [35]. In this network, the information only flows forward, i.e. there are no feedback connections which connect the outputs of previous time steps back into the network. The most basic component of an MLP is the *fully-connected* layer. In this type of layer, each neuron is connected to all of the layer's inputs An example of a feedforward neural network with a fully-connected layer is shown in Figure 2.10. Each arrow in the figure represents a connection between the output of a neuron x_i and an input of a neuron h_j, and the strength of the connection

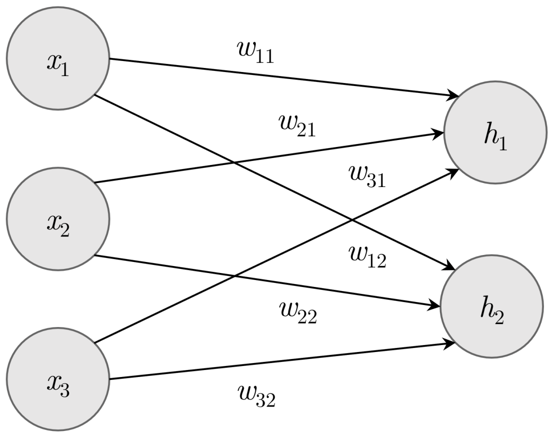


Figure 2.10: Fully-connected layer with two neurons connected to three inputs.

is represented with a weight w_{ij}. The outputs h_1 and h_2 are calculated as follows:

$$h_1 = g(w_{11}x_1 + w_{21}x_2 + w_{31}x_3) \tag{2.13}$$

$$h_2 = g(w_{12}x_1 + w_{21}x_2 + w_{31}x_3) \tag{2.14}$$

where g is a non-linear activation function such as the sigmoid function (Equation 2.6) or the rectified linear unit (ReLU) $g(x) = \max(0, x)$ [44]. In practice, it is common to add a bias term b to each non-linear transformation. More generally, a fully-connected layer with n inputs and m outputs can be written in its vectorised form as:

$$\boldsymbol{h} = g(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) \tag{2.15}$$

where the matrix of weights $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ and the vector of biases $\boldsymbol{b} \in \mathbb{R}^m$ are the parameters of the layer. Note that a feedforward neural network with a single fully-connected layer is equivalent to a linear binary classifier if there is a single output and the activation function is a sigmoid function (Equation 2.5), or to a multiclass linear classifier if there are multiple outputs and the activation function is a softmax function (Equation 2.8).

The non-linear transformation described in Equation 2.15 can be chained to form feedforward neural networks with an arbitrary number of fully-connected layers. For example, the output of a feedforward neural network with three fully-connected layers is calculated as follows:

$$\boldsymbol{h}_1 = g(\boldsymbol{W}_1^T\boldsymbol{x} + \boldsymbol{b}_1) \tag{2.16}$$

$$\boldsymbol{h}_2 = g(\boldsymbol{W}_2^T\boldsymbol{h}_1 + \boldsymbol{b}_2) \tag{2.17}$$

$$\hat{\boldsymbol{y}} = g(\boldsymbol{W}_3^T\boldsymbol{h}_2 + \boldsymbol{b}_3) \tag{2.18}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$ and $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_3$ are the weights and biases of the three layers respectively, $\boldsymbol{x}$ is the input of the network, $\hat{\boldsymbol{y}}$ is the output of the network, and $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are the outputs of the intermediate layers, also called hidden layers. Note that if a linear function is used as the activation function g, the entire neural network behaves like a linear model. By stacking layers with non-linear activation functions, neural networks are able to learn complex non-linear functions of their inputs commonly referred to as features (as they replace the traditional hand-engineered features used in other machine learning algorithms). Concretely, each layer in the network builds up on the features learnt by the preceding layer to learn features of increasing complexity. For example, in the case of images, the first layers might compute low-level features such as image gradients, whereas the last layers might compute high-level features such as the presence of particular objects in the image.

2.3.2 Convolutional Neural Networks

A *convolutional neural network* (CNN) [45, 46] is a type of feedforward neural network specifically designed to process data that can be spatially arranged such as images and videos. CNNs are mainly composed of convolutional layers. These layers differ from fully-connected layers in terms of how the neurons are connected to their inputs. In a fully-connected layer, each neuron is connected to all of the inputs, as shown above in Figure 2.10. In contrast, in a convolutional layer, each neuron is connected to a small region of neighbouring inputs called a *receptive field.* This type of connectivity pattern is known as *local connectivity* or *sparse connectivity.* The neurons in a convolutional layer work as filters that are convolved over the input image to produce *feature maps.* This transformation is analogous to the spatial convolution operation introduced in Section 2.1 but typically operates over three dimensions: width, height and depth. A feature map $\boldsymbol{H}$ is calculated by computing the dot product between the weights in a filter and the corresponding elements in an input volume, plus a bias term. Thus, at each location i, j, k in an input volume $\boldsymbol{X}$, the output of a convolutional layer is computed as follows:

$$\boldsymbol{H}_{i,j,k} = \sum_{l,m,n} g(\boldsymbol{X}_{i+l,j+m,k+n}\boldsymbol{W}_{l,m,n} + b) \tag{2.19}$$

where $\boldsymbol{W}$ is a filter, b is a bias term and g is an activation function. From this definition, it is easy to see that convolutional layers extract the same features from all the regions in an image, since the same filter $\boldsymbol{W}$ is used across all spatial locations i, j, k. This property is useful in many computer vision tasks. For example in object recognition, where the goal is to recognise an object that can appear anywhere in the image.

Since each filter computes one kind of feature, convolutional layers typically use many different filters in parallel to extract many kinds of features. Figure 2.11 shows how a $64 \times 64 \times 3$ input volume convolved with 6 filters with a receptive field of 9×9 pixels produces a $56 \times 56 \times 6$ output volume, i.e. 6 feature maps. Note that the width and height of the output volume can be preserved by zero-padding the input volume so that the filter can be applied at the borders.

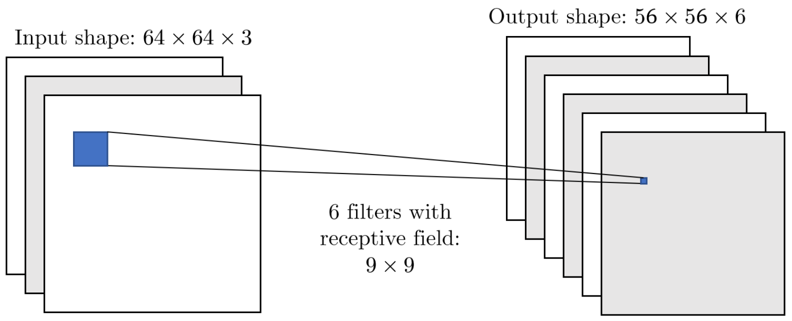

Input shape: $64 \times 64 \times 3$
Output shape: $56 \times 56 \times 6$
6 filters with receptive field: 9×9

Figure 2.11: Convolutional layer with 6 filters. The blue region in the input volume represents the receptive field. Note how the depth of the filters has to match the depth of the input volume. Also note how the spatial dimensions of the output volume shrink since the convolution operation is not defined at the borders.

The number of parameters in a convolutional layer is drastically reduced compared to a fully-connected layer since the weights of a convolutional layer are shared across all spatial locations in the input (i.e. the same filter is slid over all spatial locations in the input). In a convolutional layer, the number of weights depends on the number of filters and the dimensions of their receptive fields rather than on the number of outputs. For example, to extract 50 different features from a $100 \times 100 \times 3$ resolution image using a fully-connected layer, 150,000 weights are needed ($50 \times 100 \times 100 \times 3$). On the other hand, to extract 50 different features from a $100 \times 100 \times 3$ resolution image using a convolutional layer containing filters with a receptive field of 3×3 pixels, only 1,350 weights are needed ($50 \times 3 \times 3 \times 3$). Note that the features extracted using a fully-connected layer are global (i.e. all the inputs are used in the computation), as opposed to the local features extracted using a convolutional layer. This means that extra layers are needed in a CNN to combine local features into global features. However, due to the reduced number of parameters, CNNs can stack many more layers (and therefore extract higher-level features) than fully-connected layers without risking overfitting.

Another important component of convolutional neural networks is subsampling [46]. The idea is to reduce the size of the feature maps as more layers are stacked. In this way, the number of parameters and the computational complexity of the model are reduced, and the final representation learnt by the model becomes more compact. The subsampling operation can be implemented with *pooling* layers or with *strided* convolutions. Pooling layers are layers without parameters that simply calculate the average or the maximum value over small input regions. A strided convolution performs the same operation as a normal convolutional layer but skips some spatial locations. For example, a convolutional layer with a stride of 2 would skip half of the spatial locations and therefore produce a feature map with half the dimensions of the input.

It is also common to add fully-connected layers as the top layers of a convolutional neural network

Table 2.1: Specification of a simple CNN architecture. The input size of the network is a $32 \times 32 \times 3$ image and the output is a 10-dimensional vector. The number of filters in each convolutional layer is indicated implicitly as the output shape depth. Note that the activation function between layers has been omitted for brevity.

Layer	Filter size	Stride	Zero-padding	Output shape
Convolutional	5×5	1	Yes	$32 \times 32 \times 32$
Convolutional	3×3	1	No	$30 \times 30 \times 32$
Max pooling	2×2	2	No	$15 \times 15 \times 32$
Convolutional	3×3	1	Yes	$15 \times 15 \times 64$
Convolutional	3×3	1	No	$13 \times 13 \times 64$
Max pooling	2×2	2	No	$6 \times 6 \times 64$
Fully-connected	-	-	-	512
Fully-connected	-	-	-	10

to combine the outputs of all the feature maps and produce the final prediction output of the network. An example specification of a full CNN architecture is shown in Table 2.1.

2.3.3 Deep Generative Models

Generative models can learn an estimate p of the data generating distribution p_{data} in an unsupervised fashion. Some models are able to learn p explicitly and others can only learn to sample from p. There are three type of deep generative models that have recently gained popularity: *autoregressive models* [47], *variational autoencoders* [48] and *generative adversarial networks* [49].

Autoregressive models [47] learn the probability distribution p explicitly by decomposing the probability of an input sample $\boldsymbol{x}$ as a product of conditional probabilities using the chain rule:

$$p(\boldsymbol{x}) = \prod_i p(x_i \mid x_1, ..., x_{i-1}) \tag{2.20}$$

In a deep autoregressive model, the conditional probability of each element $p(x_i \mid x_1, ..., x_{i-1})$ is computed by a deep neural network. The main disadvantage of this approach is that during inference the samples are generated sequentially, as each element x_i depends on the previously computed elements $x_1, ..., x_{i-1}$. This sequential generation becomes very slow when a deep neural network is used to generate samples with a significant number of elements such as images [50] or sound waves [51].

Variational autoencoders [48] learn an approximation of the distribution p by maximising a lower bound $\mathcal{L}(\boldsymbol{x}) \leq \log p(\boldsymbol{x})$. In practice, this is done by maximising the probability of reconstructing the input samples using an encoder-decoder neural network and minimising the difference between the distribution learnt by the encoder and a simple prior distribution (e.g. a Gaussian distribution). Variational autoencoders are faster than autoregressive models as they can generate samples in a

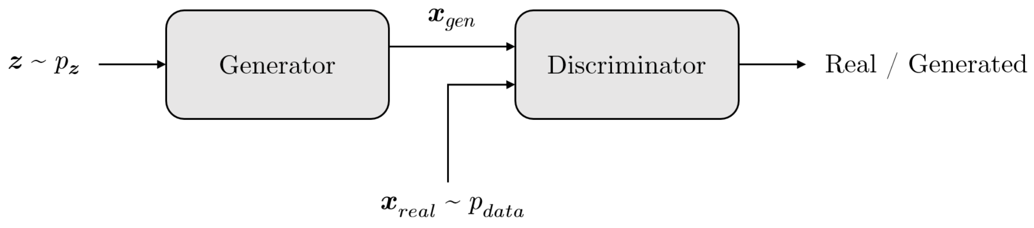

$\boldsymbol{z} \sim p_{\boldsymbol{z}}$
Generator
$\boldsymbol{x}_{gen}$
Discriminator
Real / Generated
$\boldsymbol{x}_{real} \sim p_{data}$

Figure 2.12: Vanilla generative adversarial network architecture.

single step. However, the samples generated by variational autoencoders tend to be blurry due to the mean squared error typically used in the reconstruction term.

Generative adversarial networks (GANs) [49] learn to generate samples without explicitly learning p. A vanilla GAN consists of a generator network G and a discriminator network D. The generator network G learns to map samples $\boldsymbol{z}$ from a simple probability distribution $p_{\boldsymbol{z}}$ to samples $G(\boldsymbol{z})$ that look as if they were drawn from the data generating distribution p_{data}. The discriminator network D is a binary classifier that outputs a scalar $D(\boldsymbol{x})$ representing the probability that a sample $\boldsymbol{x}$ is real rather than generated (i.e. the probability that $\boldsymbol{x}$ was sampled from p_{data} rather than generated by G). The term *adversarial* refers to the fact that the discriminator is trained to maximise the probability of assigning the correct label while the generator is trained to minimise the probability that the discriminator classifies generated samples correctly. Figure 2.12 shows the architecture of a vanilla GAN. The GAN training objective can be expressed as follows:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.21}$$

GANs are trained by optimising G and D alternatively. Since the two networks have opposing optimisation objectives, the training converges when an equilibrium is reached. GANs can generate samples in a single step and, in general, produce better looking samples than variational autoencoders. The main drawback of GANs is that their objective function is difficult to optimise in practice. Although, some alternatives such as the Wasserstein GAN [52] seem to alleviate this issue.

2.4 Conclusions

This chapter has given an overview of the most important techniques used to develop computer vision applications: feature engineering, machine learning and deep learning. The combination of hand-engineered features and traditional machine learning methods has been very successful for many years. However, in the last few years, deep learning techniques have quickly become the preferred approach for tackling complex computer vision tasks.

The background provided in this chapter will be useful to better understand the contributions made in this thesis. In particular, feature engineering and traditional machine learning techniques are used in Chapter 4; convolutional networks are used in Chapter 5; and deep generative models are used in Chapter 6.

Chapter 3

Face Recognition

Starting in the seventies, face recognition has become one of the most researched topics in computer vision and biometrics. With the availability of large databases of face images and the appropriate computational resources, current algorithms based on deep neural networks are able to recognise faces in-the-wild with near human-level accuracy. One of the reasons why face recognition has attracted so much interest over the years is the wide range of commercial and law enforcement applications. Some of these include access control, surveillance, criminal identification, advertising and identity authentication. Another reason is the possibility of deploying face recognition systems without any user interaction. Other biometric modalities such as fingerprint or iris recognition are considered more robust than face recognition but are less appealing as they require the users to cooperate with the system.

This chapter starts with an overview of face recognition, its different modes of operation, and the most popular metrics used for evaluating them. After that, an extensive literature review of the subject is provided. The content of this chapter has been adapted for an academic review article[1].

3.1 Overview

Face recognition refers to the capability of computers to correctly verify or identify face images. From a practical point of view, verification and identification are considered the two main face recognition modes of operation. Face verification refers to one-to-one matching and involves confirming the claimed identity of a subject[2]. Face identification refers to one-to-many matching and involves finding the identity of a subject in a database of enrolled subjects. Face identification systems can be further divided into closed-set and open-set. In closed-set identification systems, it is assumed

[1]An online preprint is available as *Face Recognition: From Traditional to Deep Learning Methods*, arXiv preprint arXiv:1811.00116, 2018.

[2]The term *subject* here refers to a face image of a particular subject.

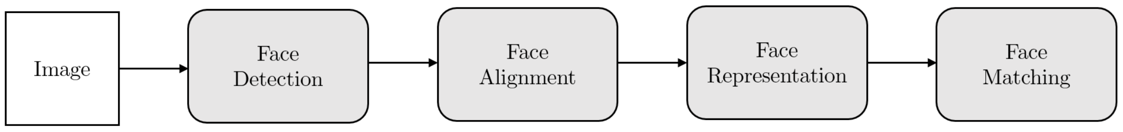


Figure 3.1: Face recognition building blocks.

that all the queried subjects are already enrolled in the system, whereas in open-set identification systems any subject can be queried.

3.1.1 Building Blocks

In practice, face verification and face identification systems are usually composed of the same building blocks. These are shown in Figure 3.1 and summarised below:

1. **Face detection**. A face detector finds the position of the faces in an image and (if any) returns the coordinates of a bounding box for each one of them. This is illustrated in Figure 3.2a.
2. **Face alignment**. The goal of face alignment is to scale and crop face images in the same way using a set of reference points located at fixed locations in the image. This process typically requires finding a set of facial landmarks using a landmark detector and, in the case of a simple 2D alignment, finding the best affine transformation that fits the reference points. Figures 3.2b and 3.2c show two face images aligned using the same set of reference points. More complex 3D alignment algorithms (e.g. [53]) can also achieve face frontalisation, i.e. changing the pose of a face to frontal.
3. **Face representation**. At the face representation stage, the pixel values of a face image are transformed into a compact and discriminative feature vector, also known as a template. Ideally, all the faces of a same subject should map to similar feature vectors. Face representation is arguably the most important component of a face recognition system and the main subject of study in this thesis (it is also the focus of the literature review in Section 3.2).
4. **Face matching**. In the face matching building block, two templates are compared to produce a similarity score that indicates the likelihood that they belong to the same subject.

Note that the generic workflow depicted in Figure 3.1 can include extra building blocks for additional functionality. For example, some face recognition systems might use multiple images of a same subject to create a template. Similarly, face verification systems might apply a threshold to the similarity score to provide a definitive *match / not match* decision. Furthermore, face identification systems typically need an extra building block to perform face matching between a query subject and all the subjects enrolled in the system. On the other hand, some face recognition systems

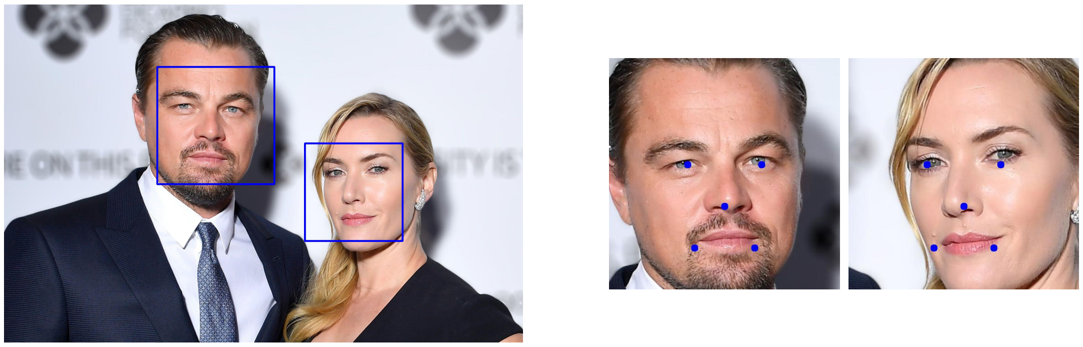


Figure 3.2: (a) Bounding boxes found by a face detector. (b) and (c) Aligned faces and reference points.

might not have all the building blocks shown in Figure 3.1. For example, face identification can be implemented by a classifier, wherein the output of the classifier is the identity of the queried subject (open-set systems can use an extra *unknown* class for non-enrolled subjects). However, this approach is less flexible since the classifier needs to be retrained every time a new subject is enrolled in the system and the model size tends to grow in proportion to the number of enrolled subjects (which can be prohibitively large in real-world deployments).

3.1.2 Evaluation Metrics

In a verification system, the comparison of two templates belonging to the same subject is known as a *positive* comparison and the comparison of two templates belonging to different subjects is known as a *negative* comparison. Thus, four outcomes are possible when two templates are compared and a threshold is applied to the resulting similarity score [54]:

- **True positive** if the result of a positive comparison is a similarity score above the threshold.
- **False negative** if the result of a positive comparison is a similarity score below the threshold.
- **True negative** if the result of a negative comparison is a similarity score below the threshold.
- **False positive** if the result of a negative comparison is a similarity score above the threshold.

See Figure 3.3 for a visual representation of how the threshold determines the number of occurrences of each outcome. Given these four possible outcomes, the performance of a face verification algorithm can be fully characterised using the *true positive rate*, also known as the *true acceptance rate* (TAR), and the *false positive rate*, also known as the *false acceptance rate* (FAR):

$$\mathrm{TAR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.1}$$

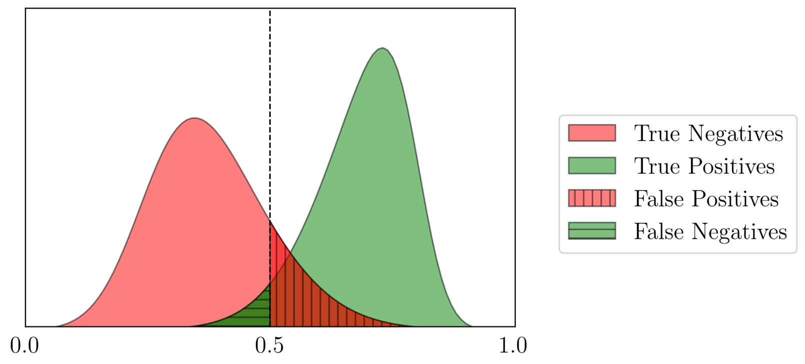


Figure 3.3: Distribution of similarity scores and a threshold of 0.5 in a verification system. Each possible outcome is indicated with a different combination of colour and pattern.

$$\text{FAR} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{3.2}$$

where P is the number of attempted positive comparisons, N is the number of attempted negative comparisons, and TP, FN, TN and FP are the number of true positives, false negatives, true negatives and false positives respectively. Alternatively, the same information can be provided using the complementary *false negative rate*, also known as the *false rejection rate* (FRR), and the *true negative rate*, also known as the *true rejection rate* (TRR):

$$\text{FRR} = 1 - \frac{\text{TP}}{\text{P}} = \frac{\text{FN}}{\text{P}} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{3.3}$$

$$\text{TRR} = 1 - \frac{\text{FP}}{\text{N}} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.4}$$

Since the TAR and FAR (and FRR and TRR) are relative to the threshold used to make the *match / not match* decision, it is common to plot them for a range of possible thresholds. The resulting plot is known as a *receiver operating characteristic* (ROC) curve [54]. Each point in a ROC curve is an *operating point* associated with a different TAR, FAR and threshold. An example ROC curve is shown in Figure 3.4. Note that the verification performance of an algorithm gets better as the curve approaches the upper left corner (i.e. high TAR at low FAR for any given threshold). Also note that the diagonal dashed line in Figure 3.4 represents random verification performance wherein the number of false positives is equal to the number of false negatives for any given threshold. The ROC curve can be summarised by reporting the TAR at a number of fixed FAR values. Sometimes, it is more convenient to report the performance with a single number. In those cases, the rate at which the FRR is equal to the FAR is commonly used. This is known as the equal error rate (EER).

The output of an identification system is often a sorted list containing the most similar identities to the identity of a queried subject. A rank-k is given when the queried subject is ranked at position k

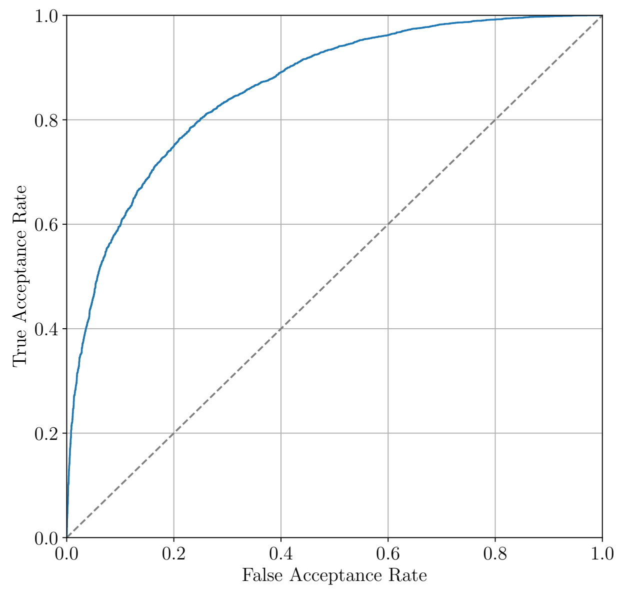


Figure 3.4: Example of a ROC curve generated using synthetic data.

by the system. The most common metric used to evaluate identification systems is the identification rate at rank-k, which indicates the probability that a subject is ranked amongst the first k positions. For example, a 90% identification rate at rank-10 means that there is a 90% probability that a queried subject will be ranked amongst the first 10 positions by the identification system.

A plot of the identification rate at different ranks, known as *cumulative match characteristic* (CMC) curve [55], is typically used to report the performance of closed-set identification systems. In open-set identification systems, a threshold is used when comparing a query subject against all the enrolled subjects to make sure that the identity exists in the database. For this reason, open-set identification systems can be seen as an hybrid between verification and closed-set identification systems. Their performance can be reported with ROC curves at different ranks or with CMC curves at different operating points. An example of closed-set and open-set CMC curves is shown in Figure 3.5.

3.2 Literature Review

Early research on face recognition focused on methods that used image processing techniques to match simple features describing the geometry of the faces. Even though these methods only worked under very constrained settings, they showed that it is possible to use computers to automatically

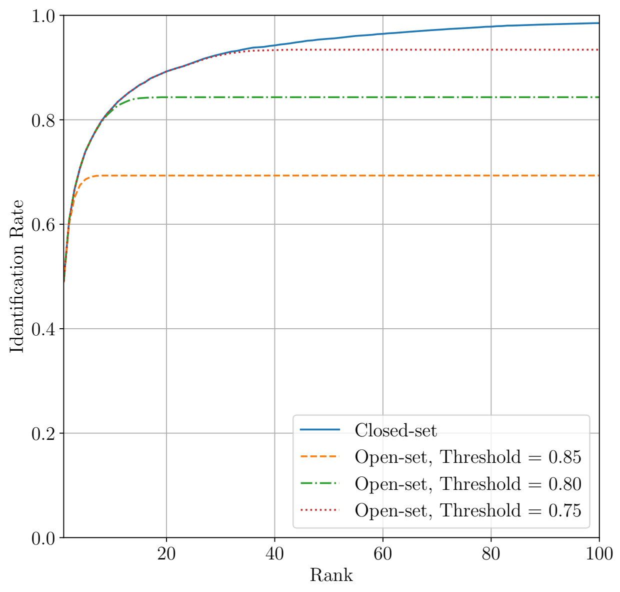


Figure 3.5: Example of closed-set and open-set CMC curves generated using synthetic data. Note that similarity scores below the thresholds are not taken into account for the computation of the open-set CMC curves.

recognise faces. After that, statistical subspaces methods such as principal component analysis (PCA) and linear discriminant analysis (LDA) gained popularity. These methods are referred to as *holistic* since they use the entire face region as an input. At the same time, progress in other computer vision domains led to the development of local feature extractors that are able to describe the texture of an image at different locations. *Feature-based* approaches to face recognition consist of matching these local features across face images. Holistic and feature-based methods were further developed and combined into *hybrid* methods. Face recognition systems based on hybrid methods remained the state-of-the-art until recently, when deep learning emerged as the leading approach to most computer vision applications, including face recognition. The rest of this section provides a summary of some of the most representative research works on each of the aforementioned types of methods.

3.2.1 Geometry-based Methods

Kelly's [1] and Kanade's [2] PhD theses in the early seventies are considered the first research works on automatic face recognition. They proposed the use of specialised edge and contour detectors to find the location of a set of facial landmarks and to measure relative positions and distances between them. The accuracy of these early systems was demonstrated on very small databases of faces (a database of 10 subjects was used in [1] and a database of 20 subjects was used in [2]). In [56], a geometry-based method similar to [2] was compared with a method that represents face images as gradient images. The authors showed that comparing gradient images provided better recognition accuracy than comparing geometry-based features. However, the geometry-based method was faster and needed less memory. The feasibility of using facial landmarks and their geometry for face recognition was thoroughly studied in [57]. Specifically, they proposed a method based on measuring the Procrustes distance [58] between two sets of facial landmarks and a method based on measuring ratios of distances between facial landmarks. The authors argued that even though other methods that extract more information from the face (e.g. holistic methods) could achieve greater recognition accuracy, the proposed geometry-based methods were faster and could be used in combination with other methods to develop hybrid methods. Geometry-based methods have proven more effective in 3D face recognition thanks to the depth information encoded in 3D landmarks [59, 60].

Geometry-based methods were crucial during the early days of face recognition research. They can be used as a fast alternative to (or in combination with) the more advanced methods described in the rest of this review.

3.2.2 Holistic Methods

Holistic methods represent faces using the entire face region. Many of these methods work by projecting face images onto a low-dimensional space that discards superfluous details and variations not needed for the recognition task. One of the most popular approaches in this category is based on PCA, discussed in Section 2.2.2. The idea, first proposed in [61, 62], is to apply PCA to a set of training face images in order to find the eigenvectors that account for the most variance in the data distribution. In this context, the eigenvectors are typically called *eigenfaces* due to their resemblance to real faces, as shown in Figure 3.6. New faces can be projected onto the subspace spanned by the eigenfaces to obtain the weights of the linear combination of eigenfaces needed to reconstruct them. This idea was used in [63] to identify faces by comparing the weights of new faces to the weights of faces in a gallery set. A probabilistic version of this approach based on a Bayesian analysis of image differences was proposed in [64]. In this method, two sets of eigenfaces were used to model intra-personal and inter-personal variations separately. Many other variations of the original eigenfaces method have been proposed. For example, a nonlinear extension of PCA based on kernel methods, namely kernel PCA [65], was proposed in [66]; independent component analysis (ICA) [67], a generalisation of PCA that can capture high-order dependencies between pixels, was proposed in [68]; and a two-dimensional PCA based on 2D image matrices instead of 1D vectors was proposed in [69].

One issue with PCA-based approaches is that the projection maximises the variance across all the images in the training set. This implies that the top eigenvectors might have a negative impact

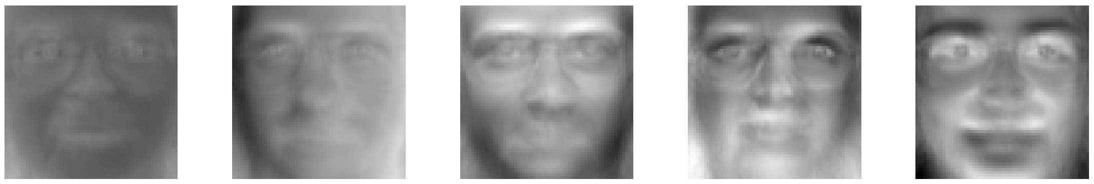


Figure 3.6: Top 5 eigenfaces computed using the ORL database of faces [70] sorted from most variance (left) to least variance (right).

on the recognition accuracy since they might correspond to intra-personal variations that are not relevant for the recognition task (e.g. illumination, pose or expression). Holistic methods based on *linear discriminant analysis* (LDA), also called *Fisher discriminant analysis*, [71] have been proposed to solve this issue [72, 73, 74, 75]. The main idea behind LDA is to use the class labels to find a projection matrix $\boldsymbol{W}$ that maximises the variance between classes while minimising the variance within classes:

$$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \frac{|\boldsymbol{W}^T \boldsymbol{S}_b \boldsymbol{W}|}{|\boldsymbol{W}^T \boldsymbol{S}_w \boldsymbol{W}|} \tag{3.5}$$

where $\boldsymbol{S}_w$ and $\boldsymbol{S}_b$ are the between-class and within-class scatter matrices defined as follows:

$$\boldsymbol{S}_w = \sum_{k}^{K} \sum_{\boldsymbol{x}_j \in C_k} (\boldsymbol{x}_j - \boldsymbol{\mu}_k)(\boldsymbol{x}_j - \boldsymbol{\mu}_k)^T \tag{3.6}$$

$$\boldsymbol{S}_b = \sum_{k}^{K} (\boldsymbol{\mu} - \boldsymbol{\mu}_k)(\boldsymbol{\mu} - \boldsymbol{\mu}_k)^T \tag{3.7}$$

where $\boldsymbol{x}_j$ represents a data sample, $\boldsymbol{\mu}_k$ is the mean of class C_k, $\boldsymbol{\mu}$ is the overall mean and K is the number of classes in the dataset. The solution to Equation 3.5 can be found by computing the eigenvectors of the separation matrix $\boldsymbol{S} = \boldsymbol{S}_w^{-1}\boldsymbol{S}_b$. Similar to PCA, LDA can be used for dimensionality reduction by selecting a subset of eigenvectors corresponding to the largest eigenvalues. Even though LDA is considered a more suitable technique for face recognition than PCA, pure LDA-based methods are prone to overfitting when the within-class scatter matrix $\boldsymbol{S}_w$ is not correctly estimated [74, 75]. This happens when the input data is high-dimensional and not many samples per class are available during training. In the extreme case, $\boldsymbol{S}_w$ becomes singular and $\boldsymbol{W}$ cannot be computed [72]. For this reason, it is common to reduce the dimensionality of the data with PCA before applying LDA [72, 74, 75]. LDA has also been extended to the nonlinear case using kernels [76, 77] and to probabilistic LDA [78].

Support vector machines (SVMs) have also been used as holistic methods for face recognition. In [79], the task was formulated as a two-class problem by training an SVM with image differences. More specifically, the two classes are the within-class difference set, which contains all the differences between images of the same class, and the between-class difference set, which contains all the differences between images of distinct classes (this formulation is similar to the probabilistic PCA proposed in [64]). In addition, [79] modified the traditional SVM formulation by adding a parameter to control the operating point of the system. In [80], a separate SVM was trained for each class. The authors experimented with SVMs trained with PCA projections and with LDA projections. It was found that this SVM approach only gives better performance compared with simple Euclidean distance when trained with PCA projections, since LDA already encodes the discriminant information needed to recognise faces.

An approach related to PCA and LDA is the *locality preserving projections* (LPP) method proposed in [81]. While PCA and LDA preserve the global structure of the image space (maximising variance and discriminant information respectively), LPP aims to preserve the local structure of the image space. This means that the projection learnt by LPP maps images with similar local information to neighbouring points in the LPP subspace. For example, two images of the same person with open and closed mouth would be mapped to similar points using LPP, but not necessarily with PCA or LDA. This approach was shown to be superior than PCA and LDA on multiple datasets. Further improvements were achieved in [82] by making the LPP basis vectors orthogonal.

Another popular family of holistic methods is based on sparse representation of faces. The idea, first proposed in [83] as *sparse representation-based classification* (SRC), is to represent faces using a linear combination of training images:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 \tag{3.8}$$

where $\boldsymbol{y}$ is a test image, $\boldsymbol{A}$ is a matrix containing all the training images and $\boldsymbol{x}_0$ is a vector of sparse coefficients. By enforcing sparsity in the representation, most of the nonzero coefficients belong to training images from the correct class. At test time, the coefficients belonging to each class are used to reconstruct the image, and the class that achieves the lowest reconstruction error is considered the correct one. The robustness of this approach to image corruptions like noise or occlusions can be increased by adding a term of sparse error coefficients $\boldsymbol{e}_0$ to the linear combination:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}_0 \tag{3.9}$$

where the nonzero entries of $\boldsymbol{e}_0$ correspond to corrupted pixels. Many variations of this approach have been proposed for increased robustness and reduced computational complexity. For example, the discriminative K-SVD algorithm was proposed in [84] to select a more compact and discriminative set of training images to reconstruct the images; in [85], SRC was extended by using a Markov random field to model the prior assumption about the spatial continuity of the occluded regions; and in [86], it was proposed to weight each pixel in the image independently to achieve better reconstructed images.

More recently, inspired by probabilistic PCA [64], the *joint Bayesian* method [87] has been proposed. In this method, instead of using image differences as in [64], a face image is represented as the sum of two independent Gaussian variables representing intra-personal and inter-personal variations. Using this method, an accuracy of 92.4% was achieved on the challenging Labeled Faces in the Wild (LFW) dataset [88]. This is the highest accuracy reported by a holistic method on this dataset.

Holistic methods have been of paramount importance to the development of real-world face recognition systems, as evidenced by the large number of approaches proposed in the literature. In the next subsection, a popular family of methods that evolved as an alternative to holistic methods, namely feature-based methods, is discussed.

3.2.3 Feature-based Methods

Feature-based methods refer to methods that leverage local features extracted at different locations in a face image. Unlike geometry-based methods, feature-based methods focus on extracting discriminative features rather than computing their geometry[3]. Feature-based methods tend to be more robust than holistic methods when dealing with faces presenting local variations (e.g. facial expression or illumination). For example, consider two face images of the same subject in which the only difference between them is that the person's eyes are closed in one of them. In a feature-based method, only the coefficients of the feature vectors that correspond to features extracted around the eyes will differ between the two images. On the other hand, in a holistic method, all the coefficients of the feature vectors might differ. Moreover, many of the descriptors used in feature-based methods are designed to be invariant to different variations (e.g. scaling, rotation or translation).

One of the first feature-based methods was the modular eigenfaces method proposed in [89], an extension of the original eigenfaces technique. In this method, PCA was independently applied to different local regions in the face image to produce sets of *eigenfeatures*. Although [89] showed that both eigenfeatures and eigenfaces can achieve the same accuracy, the eigenfeatures approach provided better accuracy when only a few eigenvectors were used.

A feature-based method that uses binary edge features was proposed in [90]. Their main contribution was to improve the Hausdorff distance that was used in [91] to compare binary images. The Hausdorff distance measures proximity between two set of points by looking at the greatest distance from a point in one set to the closest point in the other set. In the modified Hausdorff distance proposed in [90], each point in one set has to be near some point in the other set. It was argued that this property makes the method more robust to small, non-rigid local distortions. A variation of this method proposed *line edge maps* (LEMs) for face representation [92]. LEMs provide a compact face representation since edges are encoded as line segments, i.e. only the coordinates of the end points are used. A line segment Hausdorff distance was also proposed in this work to match LEMs. The proposed distance is discouraged to match lines with different orientations, is robust to line

[3]Technically, geometry-based methods can be seen as a special case of feature-based methods, since many feature-based methods also leverage the geometry of the extracted features.

displacements, and incorporates a measure of the difference between the number of lines found in two LEMs.

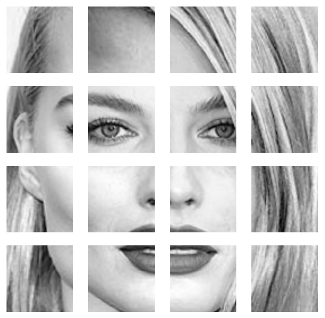
A very popular feature-based method was the *elastic bunch graph matching* (EBGM) method [93], an extension of the *dynamic link architecture* proposed in [94]. In this method, a face is represented using a graph of nodes. The nodes contain Gabor wavelet coefficients [95] extracted around a set of predefined facial landmarks. During training, a *face bunch graph* (FBG) model is created by stacking the manually located nodes of each training image. When a test face image is presented, a new graph is created and fitted to the facial landmarks by searching for the most similar nodes in the FBG model. Two images can be compared by measuring the similarity between their graph nodes. A version of this method that uses histograms of oriented gradients (HOG) [96, 97] instead of Gabor wavelet features was proposed in [98]. This method outperforms the original EBGM method thanks to the increased robustness of HOG descriptors to changes in illumination, rotation and small displacements.

With the development of local feature descriptors in other computer vision applications [99], the popularity of feature-based methods for face recognition increased. In [26], histograms of LBP descriptors were extracted from local regions independently, as shown in Figure 3.7, and concatenated to form a global feature vector. Additionally, they measured the similarity between two feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ using a weighted Chi-square distance:

$$\chi^2(\boldsymbol{a}, \boldsymbol{b}) = \sum_i \frac{w_i(a_i - b_i)^2}{a_i + b_i} \tag{3.10}$$

where w_i is a weight that controls the contribution of the i-th coefficient of the feature vectors. As shown in [100], many variations of this method have been proposed to improve face recognition accuracy and to tackle other related tasks such as face detection, facial expression analysis and demographic classification. For example, LBP descriptors extracted from Gabor feature maps, known as LGBP descriptors, were proposed in [101, 102]; a rotation invariant LBP descriptor that applies Fourier transform to LBP histograms was proposed in [103]; and a variation of LBP called *local derivative pattern* (LDP) was proposed in [104] to extract high-order local information by encoding directional pattern features.

SIFT descriptors [29] have also been extensively used for face recognition. Three different methodologies for matching SIFT descriptors across face images were proposed in [105]: (i) computing the distances between all pairs of SIFT descriptors and using the minimum distance as a similarity score; (ii) similar to (i) but SIFT descriptors around the eyes and the mouth are compared independently, and the average of the two minimum distances is used as a similarity score; and (iii) computing SIFT descriptors over a regular grid and using the average distance between the corresponding pairs of descriptors as a similarity score. The best recognition accuracy was obtained using the third method. A related method [106] proposed the use of SURF features [30] instead of


(a)

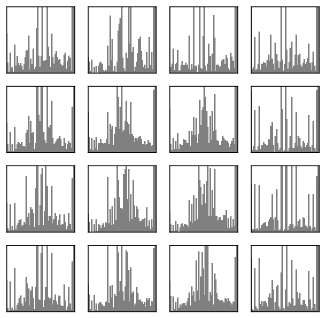

(b)

Figure 3.7: (a) Face image divided into 4 × 4 local regions. (b) Histograms of LBP descriptors computed from each local region.

SIFT. In this work, the authors observed that dense feature extraction over a regular grid provides the best results. In [107], two variations of SIFT were proposed, namely, the *volume-SIFT* which removes unreliable keypoints based on their scale, and the *partial-descriptor-SIFT* which finds keypoints at large scales and near face boundaries. Compared to the original SIFT, both approaches were shown to improve face recognition accuracy.

Some feature-based methods have focused on learning local features from training samples. For example, in [108], unsupervised learning techniques (K-means [109], PCA tree [110] and random-projection tree [110]) were used to encode local microstructures of faces into a set of discrete codes. The discrete codes were then grouped into histograms at different facial regions. The final local descriptors were computed by applying PCA to each histogram. A learning-based descriptor with similarities to LBP was proposed in [111]. Specifically, this descriptor consists of a differential pattern generated by subtracting the centre pixel of a local 3 × 3 region to its neighbouring pixels and a training of a Gaussian mixture model to compute high-order statistics of the differential pattern. Another LBP-like descriptor that has a learning stage was proposed in [112]. In this work, LDA was used to (i) learn a filter that when applied to an image enhances the discriminative ability of the differential patterns, and (ii) learn a set of weights that are assigned to the neighbouring pixels within each local region to reflect their contribution to the differential pattern.

Feature-based methods have been shown to provide more robustness to different types of variations than holistic methods. However, some of the advantages of holistic methods are lost (e.g. discarding non-discriminant information and more compact representations). Hybrid methods that combine both of these approaches are discussed next.

3.2.4 Hybrid Methods

Hybrid methods combine techniques from holistic and feature-based methods. Before deep learning became widespread, most state-of-the-art face recognition systems were based on hybrid methods. Some hybrid methods simply combine two different techniques without any interaction between them. For example, in the modular eigenfaces work [89] covered earlier, the authors experimented with a combined representation using both eigenfaces and eigenfeatures and achieved better accuracy than using either of these two methods alone. However, the most popular hybrid approach is to extract local features (e.g. LBP, SIFT) and project them onto a lower-dimensional and discriminative subspace (e.g. using PCA or LDA) as shown in Figure 3.8.

Several hybrid methods that use Gabor wavelet features combined with different subspaces methods have been proposed [113, 114, 115]. In these methods, Gabor kernels of different orientations and scales are convolved with an image and their outputs are concatenated into a feature vector. The feature vector is then downsampled to reduce its dimensionality. In [113], the feature vector was further processed using the enhanced linear discriminant model proposed in [116]. PCA followed by ICA were applied to the downsampled feature vector in [114], and the probabilistic reasoning model from [116] was used to classify whether two images belong to the same subject. In [115], kernel PCA with polynomial kernels was applied to the feature vector to encode high-order statistics. All these hybrid methods were shown to provide better accuracy than using Gabor wavelet features alone.

LBP descriptors have been a key component in many hybrid methods. In [117], an image was divided into non-overlapping regions and LBP descriptors were extracted at multiple resolutions. The LBP coefficients at each region were concatenated into regional feature vectors and projected onto PCA+LDA subspaces. This approach was extended to colour images in [118]. Laplacian PCA, an extension of PCA, was shown to outperform standard PCA and kernel PCA when applied to LBP descriptors in [119]. Two novel patch versions of LBP, namely three-patch LBP (TPLBP) and four-patch LBP (FPLBP), were combined with LDA and SVMs in [120]. The proposed TPLBP and FPLBP descriptors can boost face recognition accuracy by encoding similarities between neighbouring patches of pixels. More recently, [121] proposed a high-dimensional face representation by densely extracting multi-scale LBP (MLBP) descriptors around facial landmarks. The high-dimensional feature vector (100K-dim) was reduced to 400 dimensions by PCA and a final discriminative feature vector was learnt using joint Bayesian. In their experiments, [121] showed that extracting high-dimensional features can increase face recognition accuracy by 6-7% when going from 1K to 100K dimensions. The main drawback of this approach is the high computational costs needed to perform a dimensionality reduction of such magnitude. For this reason, they proposed to approximate the PCA and joint Bayesian transformations with a sparse linear projection matrix $\boldsymbol{B}$ by solving the following optimisation problem:

$$\min_{\boldsymbol{B}} \|\boldsymbol{Y} - \boldsymbol{B}^T \boldsymbol{X}\|_2^2 + \lambda \|\boldsymbol{B}\|_1 \tag{3.11}$$

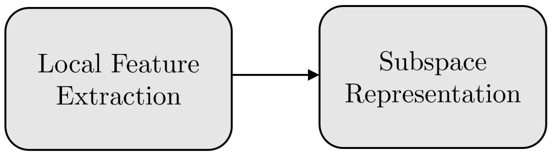


Figure 3.8: Typical hybrid face representation.

where the first term is a reconstruction error between the matrix $\boldsymbol{X}$ of high-dimensional feature vectors and the matrix $\boldsymbol{Y}$ of projected low-dimensional feature vectors; the second term enforces sparsity in the projection matrix $\boldsymbol{B}$; and λ balances the contribution of each term. Another recent method proposed a multi-task learning approach based on a discriminative Gaussian process latent variable model, named *GaussianFace* [122]. This method extended the Gaussian process approach proposed in [123] and incorporated a computationally more efficient version of kernel LDA to learn a face representation from LBP descriptors that can exploit data from multiple source domains. Using this method, an accuracy of 98.52% was achieved on the LFW dataset. This is competitive with the accuracy achieved by many deep learning methods.

Some hybrid methods have proposed to use a combination of different local features. For example, Gabor wavelet and LBP features were used in [124]. The authors argued that these two types of features capture complementary information. While LBP descriptors capture small appearance details, Gabor wavelet features encode facial shape over a broader range of scales. PCA was applied independently to the feature vectors containing the Gabor wavelet coefficients and the LBP coefficients to reduce their dimensionality. The final face representation was obtained by concatenating the two PCA-transformed feature vectors and applying a subspace method similar to kernel LDA, namely *kernel discriminative common vector* [125]. Another method that uses Gabor wavelet and LBP features was proposed in [126]. In this method, faces were represented by applying PCA+LDA to regions containing histograms of LGBP descriptors [102]. A multi-feature system was proposed in [8] to tackle face recognition under difficult illumination conditions. Three contributions were made in this work: (i) a preprocessing pipeline that reduces the effect of illumination variation; (ii) an extension of LBP, called *local ternary patterns* (LTP), which is more discriminant and less sensitive to noise in uniform regions; and (iii) an architecture that combines sets of Gabor wavelet and LBP/LTP features followed by kernel LDA, score normalisation and score fusion. A related method [127] proposed a novel descriptor robust to blur that extends *local phase quantization* (LPQ) descriptors [128] to multiple scales (MLPQ). In addition, a kernel fusion technique was used to combine MLPQ descriptors with MLBP descriptors in the kernel LDA framework. In [5], an age invariant face recognition system was proposed based on dense extraction of SIFT and multi-scale LBP descriptors combined with a novel multi-feature discriminant analysis (MFDA). The MFDA technique uses random subspace sampling [129] to construct multiple lower-dimensional feature subspaces, and bagging [130] to select subsets of training samples for LDA that contain inter-class pairs near the classification boundary to increase the discriminative ability of the representation.

To conclude this subsection, other types of hybrids methods that do not follow the pipeline described in Figure 3.8 are reviewed. In [131], low-level local features (image intensities in RGB and HSV colour spaces, edge magnitudes, and gradient directions) were used to compute high-level visual features by training attribute and simile binary SVM classifiers. The attribute classifiers detect describable attributes of faces such as gender, race and age. On the other hand, the simile classifiers detect non-describable attributes by measuring the similarity of different parts of a face to a limited set of reference subjects. To compare two images, the outputs of all the attribute and simile classifiers for both images are fed to an SVM classifier. A method similar to the simile classifiers from [131] was proposed in [132]. The main differences are that [132] used a large number of simple one-vs-one classifiers instead of the more complex one-vs-all classifiers used in [131], and that SIFT descriptors were used as the low-level features. Two metric learning approaches for face identification were proposed in [133]. The first one, called *logistic discriminant metric learning* (LDML) is based on the idea that the distance between positive pairs (belonging to the same subject) should be smaller than the distance between negative pairs (belonging to different subjects). The second one, called *marginalised kNN* (MkNN), uses a k-nearest neighbour classifier to find how many positive neighbour pairs can be formed from the neighbours of the two compared vectors. Both methods were trained on pairs of vectors of SIFT descriptors computed at fixed points on the face (corners of the mouth, eyes and nose).

Hybrid methods offer the best of holistic and feature-based methods. Their main limitation is the choice of good features that can fully extract the information needed to recognise a face. Some approaches have tried to overcome this issue by combining different types of features whereas others have introduced a learning stage to improve the discriminative ability of the features. Deep learning methods, discussed next, take these ideas further by training end-to-end systems that can learn a large number of features that are optimal for the recognition task.

3.2.5 Deep Learning Methods

Convolutional neural networks (CNNs) are the most common type of deep learning method for face recognition. The main advantage of deep learning methods is that they can be trained with large amounts of data to learn a face representation that is robust to the variations present in the training data. In this way, instead of designing specialised features that are robust to different types of intra-class variations (e.g. illumination, pose, facial expression, age, etc.), CNNs can learn them from training data. The main drawback of deep learning methods is that they need to be trained with very large datasets that contain enough variations to generalise to unseen samples. Fortunately, several large-scale face datasets containing in-the-wild face images have recently been released into the public domain [9, 10, 11, 12, 13, 14, 15] to train CNN models. Apart from learning discriminative features, neural networks can reduce dimensionality and be trained as classifiers or using metric learning approaches. CNNs are considered end-to-end trainable systems that do not need to be combined with any other specific methods.

CNN models for face recognition can be trained using different approaches. One of them consists of treating the problem as a classification one, wherein each subject in the training set corresponds to a class. After training, the model can be used to recognise subjects that are not present in the training set by discarding the classification layer and using the features of the previous layer as the face representation [134]. In the deep learning literature, these features are commonly referred to as *bottleneck* features. Following this first training stage, the model can be further trained using other techniques to optimise the bottleneck features for the target application (e.g. using joint Bayesian [9] or fine-tuning the CNN model with a different loss function [10]). Another common approach to learning face representation is to directly learn bottleneck features by optimising a distance metric between pairs of faces [135, 136] or triplets of faces [137].

The idea of using neural networks for face recognition is not new. An early method based on a probabilistic decision-based neural network (PBDNN) [138] was proposed in 1997 for face detection, eye localisation and face recognition. The face recognition PDBNN was divided into one fully-connected subnet per training subject to reduce the number of hidden units and avoid overfitting. Two PBDNNs were trained using intensity and edge features respectively and their outputs were combined to give a final classification decision. Another early method [139] proposed to use a combination of a *self-organising map* (SOM) and a convolutional neural network. A self-organising map [140] is a type of neural network trained in an unsupervised way that projects the input data onto a lower-dimensional space that preserves the topological properties of the input space (i.e. inputs that are nearby in the original space are also nearby in the output space). Note that none of these two early methods were trained end-to-end (edge features were used in [138] and a SOM in [139]), and that the proposed neural network architectures were shallow. An end-to-end face recognition CNN was proposed in [135]. This method used a *siamese* architecture trained with a contrastive loss function [141]. The contrastive loss implements a metric learning procedure that aims to minimise the distance between pairs of feature vectors corresponding to the same subject while maximising the distance between pairs of feature vectors corresponding to different subjects. The CNN architecture used in this method was also shallow and was trained with small datasets.

None of the methods mentioned above achieved groundbreaking results, mainly due to the low capacity of the networks used and the relatively small datasets available for training at the time. It was not until these models were scaled up and trained with large amounts of data [142] that the first deep learning methods for face recognition [134, 9] became the state-of-the-art. In particular, Facebook's *DeepFace* [134], one of the first CNN-based approaches for face recognition that used a high capacity model, achieved an accuracy of 97.35% on the LFW benchmark, reducing the error

Table 3.1: Public large-scale face datasets.

Dataset	Images	Subjects	Images per subject
CelebFaces+ [9]	202,599	10,177	19.9
UMDFaces [14]	367,920	8,501	43.3
CASIA-WebFace [10]	494,414	10,575	46.8
VGGFace [11]	2.6M	2,622	1,000
VGGFace2 [15]	3.31M	9,131	362.6
MegaFace [13]	4.7M	672,057	7
MS-Celeb-1M [12]	10M	100,000	100

of the previous state-of-the-art by 27%. The authors trained a CNN with softmax loss[4] using a dataset containing 4.4 million faces from 4,030 subjects. Two novel contributions were made in this work: (i) an effective facial alignment system based on explicit 3D modelling of faces, and (ii) a CNN architecture containing locally connected layers [143, 144] that (unlike regular convolutional layers) can learn different features from each region in an image. Concurrently, the *DeepID* system [9] achieved similar results by training 60 different CNNs on patches comprising ten regions, three scales and RGB or grey channels. During testing, 160 bottleneck features were extracted from each patch and its horizontally flipped counterpart to form a 19,200-dimensional feature vector ($160 \times 2 \times 60$). Similar to [134], the proposed CNN architecture also used locally connected layers. The verification result was obtained by training a joint Bayesian classifier [87] on the 19,200-dimensional feature vectors extracted by the CNNs. The system was trained on a dataset containing 202,599 face images of 10,177 celebrities [9].

There are three main factors that affect the accuracy of CNN-based methods for face recognition: training data, CNN architecture, and loss function. As in most deep learning applications, large training sets are needed to prevent overfitting. In general, CNNs trained for classification become more accurate as the number of samples per class increases. This is because the CNN model is able to learn more robust features when is exposed to more intra-class variations. However, face recognition models need to extract features that generalise to subjects not present in the training set. Hence, the datasets used for face recognition need to also contain a large number of subjects so that the model is exposed to more inter-class variations. The effect that the number of subjects in a dataset has in face recognition accuracy was studied in [145]. In this work, a large dataset was first sorted by the number of images per subject in decreasing order. Then, a CNN was trained with different subsets of training data by gradually increasing the number of subjects. The best accuracy was obtained when the first 10,000 subjects with the most images were used for training. Adding more subjects decreased the accuracy since very few images were available for each extra subject. Another study [146] investigated whether wider datasets are better than deeper datasets

[4]Here, *softmax loss* refers to the combination of the softmax activation function and the cross-entropy loss used to train classifiers.

or vice versa (a dataset is considered wider than another if it contains more subjects; similarly, a dataset is considered deeper than another if it contains more images per subject). From this study, it was concluded that given the same number of images, wider datasets provide better accuracy. The authors argued that this is due to the fact that wider datasets contain more inter-class variations and, therefore, generalise better to unseen subjects. Table 3.1 shows some of the most common public datasets used to train CNNs for face recognition.

CNN architectures for face recognition have been inspired by those achieving state-of-the-art accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). For example, a version of the *VGG* network [147] with 16 layers was used in [11], and a similar but smaller network was used in [10]. In [137], two different types of CNN architectures were explored: VGG style networks [147] and *GoogleNet* style networks [148]. Even though both types of networks achieved comparable accuracy, the GoogleNet style networks had 20 times fewer parameters. More recently, *residual* networks (ResNets) [149] have become the preferred choice for many object recognition tasks, including face recognition [150, 151, 152, 153, 154, 155, 156]. The main novelty of ResNets is the introduction of a building block that uses a shortcut connection to learn a residual mapping, as shown in Figure 3.9. The use of shortcut connections allows the training of much deeper architectures as they facilitate the flow of information across layers. A thorough study of different CNN architectures was carried out in [156]. The best trade-off between accuracy, speed and model size was obtained with a 100-layer ResNet with a residual block similar to the one proposed in [157].

The choice of loss function for training CNN-based methods has been the most recent active area of research in face recognition. Even though CNNs trained with softmax loss have been very successful [134, 9, 10, 158], it has been argued that the use of this loss function does not generalise well to subjects not present in the training set. This is because the softmax loss is encouraged to learn features that increase inter-class differences (to be able to separate the classes in the training set) but does not necessarily reduce intra-class variations. Several methods have been proposed to mitigate this issue. A simple approach is to optimise the bottleneck features using a discriminative subspace method such as joint Bayesian [87], as done in [9, 159, 160, 161, 10, 162]. Another approach is to use metric learning. For example, a pairwise contrastive loss was used as the only supervisory signal in [135, 136] and combined with a classification loss in [159, 160, 161]. One of the most popular metric learning approaches for face recognition is the triplet loss function [163], first used in [137] for the face recognition task. The aim of the triplet loss is to separate the distance between positive pairs from the distance between negative pairs by a margin. More formally, for each triplet i the following condition needs to be satisfied [137]:

$$\|f(\boldsymbol{x}_a) - f(\boldsymbol{x}_p)\|_2^2 + \alpha < \|f(\boldsymbol{x}_a) - f(\boldsymbol{x}_n)\|_2^2 \tag{3.12}$$

where $\boldsymbol{x}_a$ is an anchor image, $\boldsymbol{x}_p$ is an image of the same subject, $\boldsymbol{x}_n$ is an image of a different subject, f is a mapping learnt by a model and α is a margin that is enforced between positive and negative

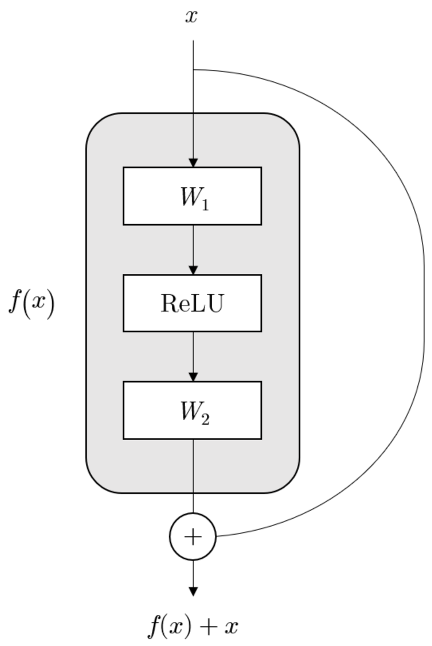
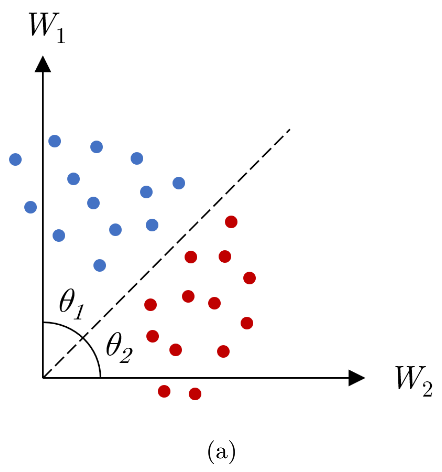


Figure 3.9: Original residual block proposed in [149]. W_1 and W_2 represent the weights of two convolutional layers respectively and ReLU is a rectifier linear unit activation function.

pairs. In practice, CNNs trained with triplet loss converge slower than with softmax loss due to the large number of triplets (or pairs in the case of contrastive loss) needed to cover the entire training set. Although this problem can be alleviated by selecting hard triplets (i.e. triplets that violate the margin condition) during training [137], it is common to train with softmax loss in a first training stage and then fine-tune bottleneck features with triplet loss in a second training stage [11, 164, 165]. Some variations of the triplet loss have been proposed. For example, in [164], the dot product was used as a similarity measure instead of the Euclidean distance, and a probabilistic triplet loss was proposed in [165]. An alternative loss function used to learn discriminative features is the centre loss proposed in [166]. The goal of the centre loss is to minimise the distances between bottleneck features and their corresponding class centres. By jointly training with softmax and centre loss, it was shown that the features learnt by a CNN could effectively increase inter-personal variations (softmax loss) and reduce intra-personal variations (centre loss). The centre loss has the advantage of being more efficient and easier to implement than the contrastive and triplet losses since it does not require forming pairs or triplets during training. Another related metric learning method is the range loss proposed in [167] for improving training with unbalanced datasets. The range loss has two components. The intra-class component of the loss minimises the k-largest distances between samples of the same class, and the inter-class component of the loss maximises the distance between the closest two class centres in each training batch. By using these extreme cases, the range loss



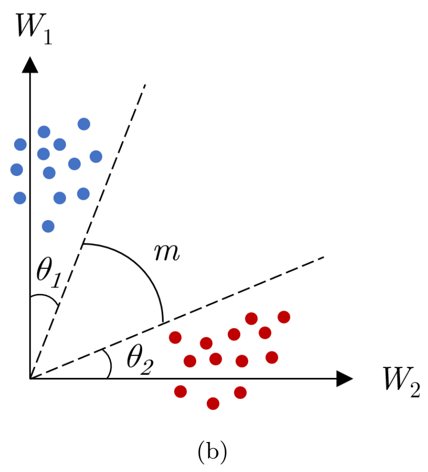


Figure 3.10: Effect of introducing a margin m in the decision boundary between two classes. (a) Softmax loss. (b) Softmax loss with margin.

uses the same information from each class, regardless of how many samples per class are available. Similar to the centre loss, the range loss needs to be combined with softmax loss to avoid the loss being degraded to zeros [166].

One of the difficulties that arise when combining different loss functions is finding the correct balance between each term. Recently, several approaches have proposed to modify the softmax loss so that it can learn discriminative features with no need to combine it with other losses. One approach that has been shown to increase the discriminative ability of the bottleneck features is feature normalisation [150, 153]. For example, [150] proposed to normalise the features to have unit L_2-norm and [153] proposed to normalise the features to have zero mean and unit variance. A very successful development has been the introduction of a margin in the decision boundary between each class in the softmax loss [168]. For simplicity, consider binary classification with softmax loss. In this case, the decision boundary between each class (if the biases are zero) is given by:

$$\|\boldsymbol{x}\|(\|\boldsymbol{W}_1\|\cos\theta_1 - \|\boldsymbol{W}_2\|\cos\theta_2) = 0 \tag{3.13}$$

where $\boldsymbol{x}$ is a feature vector, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the weights corresponding to each class and θ_1 and θ_2 are the angles between $\boldsymbol{x}$ and $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ respectively. By introducing a multiplicative margin m in Equation 3.13, the two decision boundaries become more stringent:

$$\|\boldsymbol{x}\|(\|\boldsymbol{W}_1\|\cos m\theta_1 - \|\boldsymbol{W}_2\|\cos\theta_2) = 0 \text{ for class 1} \tag{3.14}$$

$$\|\boldsymbol{x}\|(\|\boldsymbol{W}_1\|\cos\theta_1 - \|\boldsymbol{W}_2\|\cos m\theta_2) = 0 \text{ for class 2} \tag{3.15}$$

As shown in Figure 3.10, the margin can effectively increase the separation between classes and their

Table 3.2: Decision boundaries for different variations of the softmax loss with margin. Note that the decision boundaries are for class 1 in a binary classification case.

Loss function	Decision boundary
Softmax with multiplicative angular margin [151]	$\|\boldsymbol{x}\|(\cos m\theta_1 - \cos\theta_2) = 0$
Softmax with additive cosine margin [154, 155]	$s(\cos\theta_1 - m - \cos\theta_2) = 0$
Softmax with additive angular margin [156]	$s(\cos(\theta_1 + m) - \cos\theta_2) = 0$

intra-class compactness. Several alternative approaches have been proposed depending on how the margin is incorporated into the loss [151, 154, 155, 156]. For example, in [151] the weight vectors were normalised to have unit norm so that the decision boundary only depends on the angles θ_1 and θ_2. In [154, 155], an additive cosine margin was proposed. Compared to the multiplicative margin [168, 151], the additive margin is easier to implement and optimise. In this work, apart from normalising the weight vectors, the feature vectors were also normalised and scaled as done in [150]. An alternative additive margin was proposed in [156] which keeps the advantages of [154, 155] but has a better geometric interpretation since the margin is added to the angle and not to the cosine. Table 3.2 summarises the decision boundaries for the different variations of the softmax loss with margin. These approaches are the current state-of-the-art in face recognition.

Face recognition systems based on CNNs have become the standard due to the significant accuracy improvement achieved over other types of methods. Moreover, it is straightforward to scale-up these systems to achieve even higher accuracy by increasing the size of the training sets and/or the capacity of the networks. However, collecting large amounts of labelled face images is expensive, and very deep CNN architectures are slow to train and deploy. Generative adversarial networks (GANs), discussed in Section 2.3.3, are a promising solution to the first issue. Recent works on GANs with face images include facial attributes manipulation [169, 170, 171, 172, 173, 174, 175, 176, 177, 178], facial expression editing [179, 180, 174], generation of novel identities [181], face frontalisation [182, 183] and face ageing [184, 185]. It is expected that these advancements will be used to generate additional training images without requiring millions of face images to be labelled. To address the second issue, more efficient architectures such as MobileNets [186, 187] are being developed and used for real-time face recognition on devices with limited computational resources [188].

3.3 Conclusions

This chapter has provided an extensive literature review of face recognition methods divided into five main groups: geometry-based, holistic, feature-based, hybrid and deep learning methods. Although deep learning methods are the current state-of-the-art for face recognition in-the-wild, hybrid methods that combine techniques from geometry-based, holistic and feature-based methods can be useful when the face recognition task is more constrained and the computational resources are limited.

Some of the methods presented in this chapter have been used to develop the ideas in Chapters 4 to 6. More specifically, Chapter 4 proposes a novel hybrid method that makes use of SIFT descriptors, distances between facial landmarks, PCA and LDA. Chapter 5 proposes improvements to CNN-based face recognition methods. In particular, the CNN architecture proposed in [10] was used for all the experiments in Chapter 5. Finally, the state-of-the-art CNN architecture from [150, 153, 152] was used to train the discriminative models for the experiments with augmented datasets in Chapter 6.

Chapter 4

Shape and Texture Combined Face Recognition for Detection of Forged ID Documents

This chapter presents a novel hybrid face recognition method that combines texture and shape features together with subspace representation techniques. The proposed method can be used to match a face image scanned from an identity (ID) document against a face image stored in the biometric chip of such a document. The purpose of this specific face recognition task is to aid the automatic detection of forged ID documents wherein the photographic face image printed on the ID document's surface has been altered or replaced. In addition, the robustness of the proposed method when dealing with more general face recognition tasks has been proven with the Good, the Bad & the Ugly (GBU) dataset, a challenging dataset containing frontal faces. The proposed algorithm has been complemented with a novel method that adopts two operating points to enhance the reliability of the algorithm's final verification decision.

At the time this work was done, deep learning methods for face recognition were starting to gain popularity because of their increased accuracy compared to most hybrid methods. However, the focus of this work was to develop a lightweight algorithm that could be trained quickly and without a very large amount of data.

The work presented in this chapter was done during the first year of this PhD project (2015) and published in a conference proceedings in 2016 [16]. The text has been adapted from [16] with some minor modifications to better fit the style, terminology and content of this thesis. The proposed algorithm was integrated into a commercial document verification solution, both in desktop (Windows and Linux) and mobile (Android and iOS) platforms.

4.1 Introduction

This work investigates the applicability of face recognition to the authentication of biometric ID documents, i.e., the process of analysing an ID document in order to prove its legitimacy. ID documents are specifically designed with security features (watermarks, holograms, special materials, etc.) to avoid counterfeiting and forgery. A counterfeit document is a complete reproduction of a document to resemble an officially issued document, whereas a forged document is a genuine document that has been altered in some way for deceiving purposes.

Some organisations have studied how to detect fraudulent identity documents by examining their security features [189, 190]. This can be done manually by human operators, or automatically by processing the document electronically. In practice, the most effective method is a combination of both: documents are electronically validated and referred to a human operator when the automatic validation fails for any reason. Nowadays, millions of ID documents are manually examined by human operators every day in different businesses and organisations. Thus, this work is motivated by the growing need for automatic methods for authenticating ID documents. This work introduces an additional automatic validation check to detect forged documents by using face recognition for the comparison of the face image printed on the ID document to the face image stored in the ID document's biometric chip. This validation check aims to reduce a common method of forgery, namely photography substitution [189, 190].

The problem presented here is a specific face recognition case since it consists of deciding whether two images are originated from the same camera shot. Hence, the main difficulties arise from the presence of watermarks, holograms, reflections and other imperfections present on the scanned image, as shown in Figure 4.1. Other studies [191, 192] have investigated the problem of matching degraded face images scanned from passports to high-resolution digital face images. In these studies, different preprocessing methods have been proposed to improve the quality of the scanned face images for their later use in a face recognition system. While these preprocessing techniques are likely to improve the overall recognition accuracy when dealing with scanned face images, they are intrinsically domain specific. By not applying any domain specific preprocessing, the proposed method is more generalised and can be used to compare any two face images. In order to test its accuracy for both the specific task considered here as well as more general face recognition tasks, the proposed method has been evaluated on two different datasets, namely (i) a proprietary dataset (hereinafter referred to as the BiometricID dataset) containing face images scanned from ID documents and digital images from the biometric chip of those documents; and (ii) the Good, the Bad & the Ugly (GBU) dataset, which contains pairs of frontal face images with three levels of difficulty [193].

The proposed face recognition method is based on the fusion of texture features and shape features. In particular, SIFT descriptors are used as texture features, and a set of coefficients that represent relative distances between pairs of facial landmarks is used to describe the shape of the face. These two different types of features are further processed using PCA and LDA to project

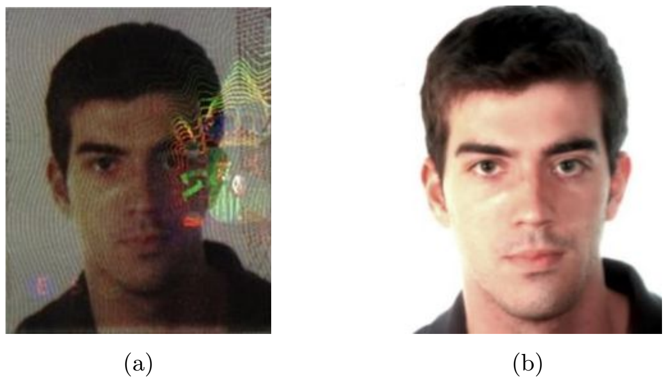


Figure 4.1: (a) Image printed on an ID document. (b) Image stored in the biometric chip of the same ID document.

the features onto a lower dimensional and more discriminative space. Furthermore, the use of two operating points is proposed to provide a greater degree of control over the verification decision.

The remainder of this chapter is structured as follows. Section 4.2 provides a subject review with a focus on the approaches adopted in this work. Section 4.3 describes the proposed method in detail and Section 4.4 presents and analyses the experimental results. Finally, conclusions are drawn in Section 4.5.

4.2 Subject Review

4.2.1 Face Alignment with Constrained Local Neural Field

As explained in Section 3.1.1, the term face alignment is typically used in the literature to refer to the automatic detection of facial landmarks in order to normalise a face image to a canonical view using 2D or 3D transformations. In this work, the landmarks are also used to compute shape features. In particular, the proposed algorithm uses the *constrained local neural field* (CLNF) [194] method for landmark detection, an extension of the *constrained local model* (CLM) proposed in [195]. Both CLM and CLNF are based on the popular *active appearance model* (AAM) proposed in [196]. An AAM is a statistical model that combines a shape model and a texture model created from a training set of images with manually annotated landmarks. After training, an AAM can be fitted to new unseen images. The fitting process starts by placing landmark points on an image using the mean location of each landmark in the training set. Then, the texture residual between the intensity of the pixel values in the image and the intensity of the pixel values obtained with the current model parameters is computed, and the model parameters are updated in order to minimise that residual. In both CLM and AAM methods, the shape of an image is modelled using a set of rigid and non-rigid shape parameters $\boldsymbol{p} = [s, \boldsymbol{R}, \boldsymbol{t}, \boldsymbol{q}]$ that models the positions of a predefined set of

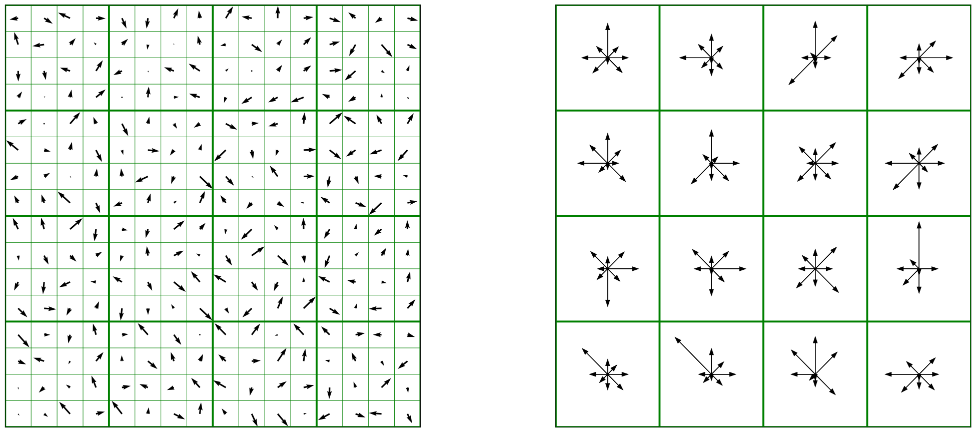

(a) (b)

Figure 4.2: (a) Image gradients computed over a 16 × 16 pixel region. (b) Orientation histograms computed over 4 × 4 pixel subregions using the gradients from (a).

landmarks:

$$\boldsymbol{x} = s\boldsymbol{R}(\bar{\boldsymbol{x}} + \boldsymbol{\Phi q}) + \boldsymbol{t} \tag{4.1}$$

where $\boldsymbol{x}$ are the locations of a set of landmarks, $\bar{\boldsymbol{x}}$ is the mean of the shapes in a training set, $\boldsymbol{\Phi}$ is a matrix of principal components describing the non-rigid modes of variation within the shapes in a training set, and $\boldsymbol{q}$ is a vector of weights that control the non-rigid shape. The rigid shape transformations are controlled by a scaling term s, a translation term $\boldsymbol{t}$, and a rotation matrix $\boldsymbol{R}$.

CLM-based methods offer better performance than AAM by using local descriptors to represent the texture surrounding each landmark location instead of using a global texture model of the whole face. Specifically, the CLNF method uses a local descriptor based on a neural network with one hidden layer, and similarity and sparsity spatial constraints in the output to enhance the landmark location accuracy [194].

4.2.2 Scale-Invariant Feature Transform

SIFT descriptors have been extensively used in face recognition, as discussed in Section 3.2.3. The original SIFT algorithm [29] finds keypoints in an image and computes a descriptor for each one of those keypoints. This section summarises how to compute the descriptor. The process of finding the keypoints is not discussed here since the method proposed in Section 4.3 computes descriptors at fixed locations within an image.

The first step to compute a SIFT descriptor is to compute gradients in a local region of 16 × 16 pixels around the centre of a keypoint. Then, the magnitude of each gradient is weighted using a Gaussian weighting function to avoid abrupt changes in the descriptor and to give less emphasis to gradients located far from the centre. Orientation histograms of 8 bins are then created using gradients within 4×4 pixel subregions. In an orientation histogram the bins can be represented by vectors pointing in different directions, where the length of each vector is the magnitude of the corresponding bin, as illustrated in Figure 4.2. The final descriptor is formed by concatenating the magnitudes of the bins of all the orientation histograms in the 16×16 pixel region. Since there are 4×4 histograms, each containing 8 bins, the dimensionality of the descriptor is 128 [29].

4.2.3 PCA and LDA for Face Recognition

The PCA-based eigenfaces method, covered in Section 3.2.2, was one of the first successful face recognition techniques [63]. Although its performance has since been surpassed by more advanced approaches, PCA is still a relevant technique used in many modern face recognition systems. One of the limitations of PCA is that it does not use class labels, meaning that faces from the same identity and faces from different identities are treated in the same way. Hence, the transformation learnt by PCA is not necessarily discriminative for the task of recognising faces.

As discussed in Section 3.2.2, LDA is a similar approach to PCA but uses the class (identity) labels to find a projection that minimises the variation within classes while maximising the variation between classes, i.e., the identity labels are used to reduce intra-personal variations while increasing inter-personal variations. In this work, PCA+LDA transformations are used to avoid cases in which the within-class scatter matrix is not well estimated [72, 74, 75].

4.3 Proposed Method

4.3.1 Face Normalisation

The Viola-Jones object detector [197] is applied to detect the position of the face(s) in a given image. In order to perform face alignment, the CLNF landmark detector is used to locate a set of 68 landmarks in each face [194]. Once the landmarks have been located, the positions of the pupils (which are found at the intersecting point defined by the landmarks surrounding the eye sockets) are used to normalise the face images to a common scale (128×128 pixels) and crop area, with the eyes located at fixed locations. Lastly, the image is converted to greyscale.

4.3.2 Face Representation

In the proposed method, face images are represented in two steps: firstly, a number of features are computed to extract relevant information that is more informative than the raw pixel values; secondly, the extracted features are transformed using subspace representation methods to create feature vectors that are more discriminative and, ideally, unique to each person.

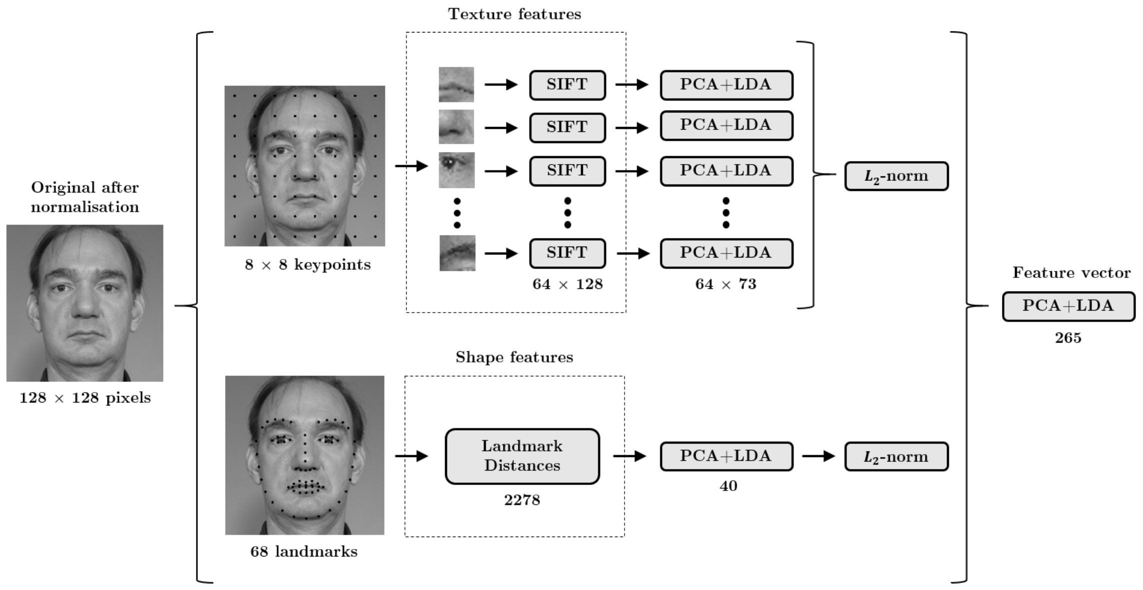


Figure 4.3: Diagram illustrating the proposed face representation method. The numbers below each block indicate the number of dimensions after each processing step.

Feature Extraction

Two different types of features are extracted from the face images in the proposed method: texture features and shape features.

Texture features are the most popular kind of features for face recognition. They can represent more information than shape features as they are directly computed from raw pixel values. Shape features are usually computed from landmark locations within a face image, which means that their reliability depends heavily on the accuracy of the landmark localisation algorithm. On the other hand, shape features can improve robustness in situations when the appearance of a face changes but not its geometry. For example, when comparing face images of the same person with and without facial hair, makeup, or glasses. In this work, shape features are fused with texture features for increased recognition accuracy.

Inspired by [198, 5], the texture features used in the proposed method consist of SIFT descriptors densely extracted from local patches in a face image. In particular, 64 SIFT descriptors are extracted from an 8×8 grid of fixed keypoints with a radius of 6 pixels, as shown in the top half of Figure 4.3.

The shape features used in the proposed method are Euclidean distances between pairs of landmarks. The distances are computed after normalising the face image so that the effect of scaling is removed. The total number of possible distances is given by $\binom{n}{2} = \frac{n(n-1)}{2}$, where n is the total number of landmarks. As shown in the bottom half of Figure 4.3, 68 landmarks are used resulting in a total of 2,278 landmark distances.

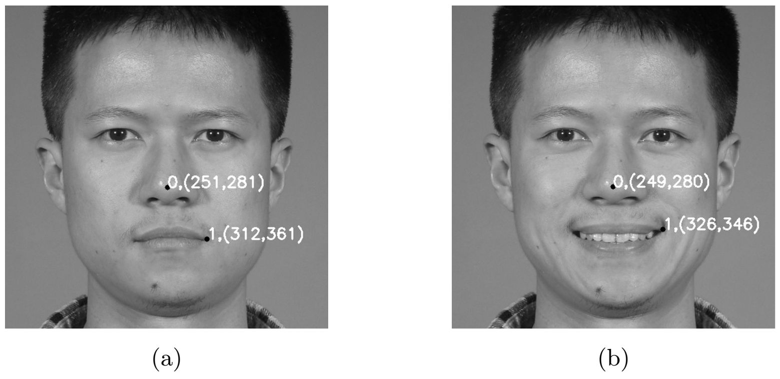


Figure 4.4: Difference between features based on distances between pairs of landmarks and features based on landmark locations. The Euclidean distance between the highlighted landmarks is 100.60 and 101.41 in (a) and (b) respectively, i.e. the relative distance between the landmarks remains almost the same in both face images. On the other hand, the misalignment of the landmark in the right corner of the mouth between (a) and (b) is 20.52. This indicates that using distances between landmarks provides extra robustness to facial expressions compared to using landmark locations. (a) Neutral facial expression. (b) Smiling facial expression.

Other studies have considered the use of landmarks for face representation. For example, the use of landmark coordinates and distances between pairs of landmarks has been studied in [57]. Their ratio-based model is very similar to the approach adopted in this work. However, they made use of fewer landmark points and removed half of the distances due to the similarity between both sides of the face. In contrast, the approach adopted in this work uses all the possible landmark points to represent as much shape information as possible. On the other hand, their model based on landmark locations is less robust to facial expressions than a distance-based model, as illustrated in Figure 4.4.

Subspace Representation

As suggested in Section 4.2.3, the proposed method uses a common technique in face recognition wherein LDA is applied to the subspace obtained by first applying PCA to the input data. In this work, PCA+LDA projections are used to independently transform the space spanned by each SIFT descriptor and the space spanned by the shape features. Other studies [5, 199, 200] have used a random sampling technique [201] to reduce the dimensionality of very high-dimensional feature spaces (e.g. the space that would result from concatenating all the shape and texture features). However, independently applying PCA+LDA to lower dimensional spaces as proposed here has the advantage of eliminating the need of such random sampling techniques, which might otherwise accidentally remove highly discriminative features.

Using a PCA that retains 98% of the original variance, the 128-dimensional SIFT descriptors become (on average) 73-dimensional when the FRGC dataset is used for training (see Section 4.4.2), and the 2,278-dimensional vector representing the landmark distances becomes 40-dimensional. The observed significant reduction in dimensionality of the shape features, notwithstanding the retention of 98% of the variance in the data, substantiates the notion that many distances between pairs of landmarks must be highly correlated as they are computed from landmarks being located close to each other.

The next step is to combine the shape and texture features. First, all the PCA+LDA projected texture features are concatenated resulting in a 4,670-dimensional vector (64 $\times$ 73) and normalised to unit L_2-norm. The PCA+LDA projected shape features are normalised in the same way prior to being concatenated with the texture features. The feature vector resulting from concatenating the shape and texture features is 4,710-dimensional (4,670 texture features and 40 shape features) and might contain redundant information. For this reason, another PCA+LDA transformation is performed to project the information contained in the concatenated texture and shape features onto a more discriminative feature space with even lower dimensionality. In this case a PCA retaining 90% of the variance is performed to reduce the dimensionality to 265 before applying LDA. A diagram describing the entire face representation step is depicted in Figure 4.3.

Another way of fusing multiple features such as texture and shape is to use score-level fusion, i.e. each feature is used in a separate algorithm and their scores are combined at the end of the process. However, feature-level fusion has the advantage of eliminating the need to optimise the fusion weighting. Liu and Wechsler [202] used a feature-level fusion method similar to the one proposed here. However, the input features in [202] are the raw image pixels and the landmark coordinates instead of the more informative features considered here.

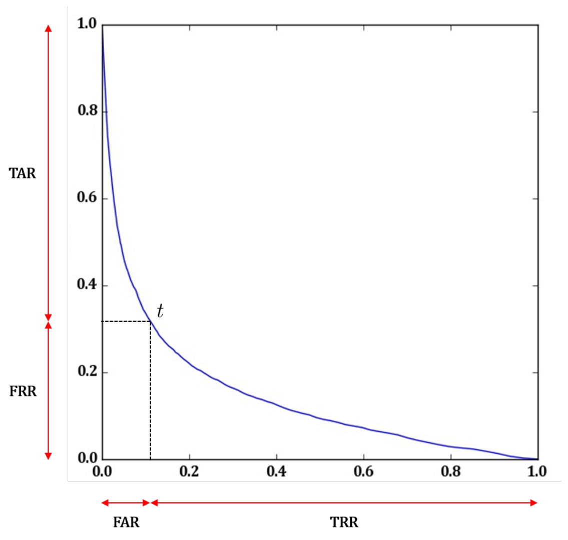
Face Matching

In this work, the matching score between two feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is computed using the cosine similarity:

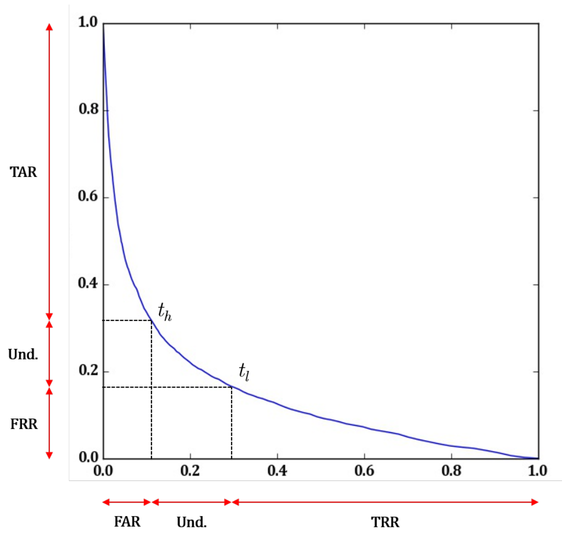
$$\cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|} \tag{4.2}$$

Typically, the operating point of a face recognition system is determined by a threshold t used to decide whether two face images match. This threshold defines the true acceptance rate and the false acceptance rate (TAR and FAR), and the true rejection rate and the false rejection rate (TRR and FRR). When analysing a conventional biometric system that operates with one threshold only, the verification accuracy is completely specified by the pair of error rates FAR and FRR, since TAR = 1 − FRR and TRR = 1 − FAR. As shown in Figure 4.5a, with one threshold (i.e. one operating point), the FRR can be reduced arbitrarily at the expense of increasing the FAR and vice versa.

In order to alleviate the conflict between the FRR and the FAR, an approach that uses two thresholds (i.e. two operating points) is adopted here. In this approach, the higher threshold t_h controls the positive rates (TAR and FAR), whereas the lower threshold t_l controls the negative



(a)



(b)

Figure 4.5: ROC curves showing the effect that modifying the threshold(s) has on the recognition rates. (a) One threshold. (b) Two thresholds. Observe how the introduction of a lower threshold t_l allows to reduce the FRR at the expense of reducing the TRR and having comparisons in which the result is undetermined. In real applications this is often desirable, as the number of negative comparisons is usually much lower than the number of positive comparisons.

rates (TRR and FRR). A matching score in between the two thresholds would yield an undetermined result. As shown in Figure 4.5b, this approach allows both the FRR and the FAR to be reduced at the same time, as they are determined separately by two different thresholds. In real applications, it is often desirable to achieve very low error rates while it is acceptable to have a certain number of undetermined cases, which might be handled further in a specific way (e.g. by manual inspection). Therefore, the introduction of a second threshold represents a significant improvement in terms of the applicability of face verification in real scenarios.

4.4 Experiments

This section presents details about the protocol and the datasets adopted for the evaluation of the proposed method (Section 4.4.1), the subsets used for training (Section 4.4.2), and the evaluation results (Section 4.4.3).

4.4.1 Evaluation Protocols and Datasets

The proposed method has been evaluated following the protocol defined by NIST for its face recognition challenges [203]. In this protocol, all the images in a target set are compared to all the images in a query set. The resulting scores are used to generate a ROC curve that plots the TAR against the FAR.

Two different datasets have been used for the evaluation. The BiometricID dataset is a proprietary dataset that contains face images scanned from ID documents (i.e. scanned images presenting artefacts such as watermarks, holograms, etc.) and the original digital face images obtained from the biometric chip of those ID documents (i.e. RFID images). For some subjects there are more than one scanned images (as the artefacts can vary from one scan to the next), and some others do not have any scanned or RFID image. In such cases, all the images are used to form additional positive or negative pairs. For example, if a subject only has one scanned image and no RFID image, the scanned image is compared against all the RFID images in the evaluation set to produce extra negative pairs. In total, the target set contains 4,802 RFID images and the query set 8,801 scanned images forming 6,029 positive pairs and 42,256,373 negative pairs.

The second dataset is the Good, the Bad and the Ugly public dataset [193], a challenging dataset used to evaluate the performance of face recognition methods on frontal images. The GBU dataset is divided into three partitions. The Good partition consists of face pairs easy to match, the Bad partition consists of face pairs with an average matching difficulty and the Ugly partition consists of face pairs difficult to match. The query and target sets of each partition contain 1,085 images from 437 subjects. The number of images per subject in each query and target set is the same in the three partitions. In total, there are 3,297 positive pairs and 1,179,928 negative pairs in each partition.

4.4.2 Training

Two different sets of images have been used to train the proposed method, one for each dataset evaluated in this work. One is a random subset of the BiometricID dataset for evaluation on the BiometricID dataset, and the other is a random subset of the FRGC dataset [204] for evaluation on the GBU dataset. Since the GBU dataset does not provide training images, the FRGC dataset is used for training, as both of them contain images that were collected by the University of Notre Dame under similar conditions.

The BiometricID training set contains 1,000 images from 500 subjects, with 2 images per subject, one RFID image and one scanned image. It is enough to use 2 images per subject since any additional sample available for a subject is simply a different scan of the face image printed on the ID document. The FGRC training set contains 5,320 images from 266 subjects, with 20 images per subject. Recognition of face images on the BiometricID dataset is considerably easier than recognition on the GBU dataset in the sense that the algorithm requires less training data to perform well on the BiometricID dataset. This is because for the face images in the BiometricID dataset, the PCA+LDA transformations only need to learn feature vectors to match two different versions of the same face image (RFID and scanned) and differentiate them from feature vectors extracted from face images of other subjects. On the other hand, the more general scenario presented by the

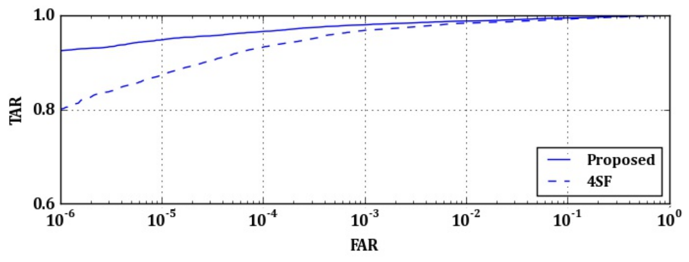


Figure 4.6: ROC curves on the BiometricID dataset using the 4SF algorithm and the proposed algorithm.

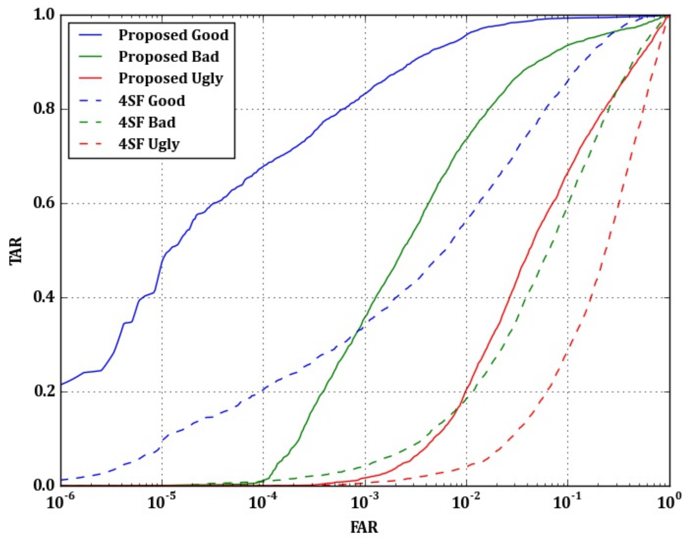


Figure 4.7: ROC curves on the Good, Bad and Ugly partition of GBU dataset using the 4SF algorithm and the proposed algorithm.

GBU dataset requires similar feature vectors extracted from various, and sometimes dramatically different, face images of each subject in the dataset.

4.4.3 Results

The open source implementation of the 4SF algorithm described in [198] has been chosen as the baseline method for the performance evaluation in this work. The 4SF algorithm is a good example of a face recognition method that makes use of local texture descriptors and subspace representation. To focus the comparison on face recognition accuracy rather than on face detection accuracy, the same face alignment technique has been adopted in the two compared methods. For this reason, the 4SF algorithm was modified to use the CLNF landmark detector to locate the position of the eye pupils. Moreover, the 4SF algorithm was trained and evaluated using exactly the same images as the proposed method.

As seen in the ROC curves obtained with the BiometricID dataset (Figure 4.6) and the GBU dataset (Figure 4.7), the proposed method outperforms the baseline in all cases. This implies that (i) the proposed combination of texture and shape features encodes more discriminative information than the texture features alone (at least when evaluating datasets that contain mainly frontal images), and (ii) the proposed PCA+LDA applied to multiple low dimensional spaces is a more effective dimensionality reduction strategy than the random sampling technique used in 4SF.

The results obtained on the GBU dataset provide a good example in which the use of two thresholds might be beneficial. For example, if only one threshold at FAR $= 10^{-3}$ was used for the *Proposed Good* system in Figure 4.7, the FRR would be approximately 0.17. Instead, if a second threshold at FAR $= 10^{-1}$ was added, the FRR would be reduced to below 0.1. The downside of adding a second threshold is having comparisons with an undetermined outcome whenever a similarity score falls between the two thresholds. However, in some applications this might be acceptable. For example, instead of replacing human decision-making entirely, this method could be used to let a face recognition algorithm handle the straightforward decisions while allowing humans to focus on the more ambiguous cases. In this way, the human workload can be reduced and the error rates can be kept relatively low.

4.5 Conclusions

This chapter presents a novel face recognition method tailored for a specific application scenario of face recognition that involves face images with added security features such as watermarks and holograms for the detection of forged ID documents.

Considering the similar face geometry across faces of the same subject, the proposed method fuses shape features with the commonly used texture features. The high dimensionality produced by the large number of shape and texture features has been reduced by using multiple PCA+LDA transformations. The proposed method has achieved high accuracy for the specific application scenario considered here. In addition, the robustness of the proposed method for more generic face recognition tasks has been confirmed by its performance on the GBU dataset. Furthermore, the proposed method offers a good balance between training and accuracy in the sense that there is no need to train using many thousands of images as is the case with the latest face recognition methods based on deep learning [134].

Finally, it has been shown how the applicability of any face recognition method in real scenarios can be benefited by the use of two thresholds to attain better control of face recognition rates at the expense of having comparisons in which the result is undetermined.

The major shortcoming of the face representation method presented in this chapter is its inability to perform well when dealing with non-frontal faces. This is because the texture and shape features used in the proposed method are only reliable when the two face images being compared are perfectly aligned (the SIFT features must be computed at the same locations and the distances between landmarks must remain constant in both face images). Moreover, the chosen feature representation is not robust to other factors that are commonly seen in faces in-the-wild such as occlusions or strong changes in facial expression and lighting. The next chapter focuses on convolutional neural networks, which are more suitable for the recognition of these types of face images.

Chapter 5

Enhancing Convolutional Neural Networks for Face Recognition with Occlusion Maps and Batch Triplet Loss

As mentioned in Chapter 3, convolutional neural networks are the state-of-the-art technique for face recognition since they can learn highly robust features from large datasets of faces. This chapter presents two novel contributions to the task of unconstrained face recognition using convolutional neural networks. Firstly, the problem of face recognition with partial occlusions is considered and it is shown how current approaches might suffer significant performance degradation when dealing with this kind of face images. A simple method is proposed to find out which parts of the human face are more important to achieve a high recognition rate. That information is then used during training to force a convolutional neural network to learn discriminative features from all the face regions more equally, including those to which typical approaches tend to pay less attention. The accuracy of the proposed method when dealing with real-world occlusions is evaluated using the AR face database. Secondly, a novel loss function called *batch triplet loss* that improves the performance of the triplet loss is proposed. This is done by adding an extra term to the loss function to minimise the standard deviation of both positive and negative scores. Consistent improvement in the Labeled Faces in the Wild (LFW) benchmark is shown by applying both proposed adjustments to the convolutional neural network training.

This work was done between the second and the third year of this PhD project (2016 - 2017) and published as a journal article in 2018 [17]. Sections 5.1 and 5.2 of this chapter have been adapted from the journal article to avoid overlap with the content of Chapter 3. The rest of the sections are almost identical to the journal article.

5.1 Introduction

Most of the recent research in the face recognition field has focused on unconstrained face recognition. As discussed in Section 3.2.5, CNN models have shown excellent performance on this task, as they are able to extract features that are robust to variations present in the training data (if enough samples containing these variations are provided). Nonetheless, this work shows how partial facial occlusions remain a problem for unconstrained face recognition. This is because most databases used for training do not present enough occluded faces for a CNN to learn how to deal with them. Common sources of occlusion include sunglasses, hats, scarves, hair, facial hair, or any object between the face and the camera. This is of particular relevance to applications where the subjects are not expected to be co-operative (e.g. security applications). One way of overcoming this problem is to train CNN models with datasets that contain more occluded faces. However, this task can be challenging because the main source of face images is usually the web, where labelled faces with occlusions are less abundant.

Bearing this in mind, this chapter proposes a novel data augmentation approach for generating occluded face images in a strategic manner. A technique similar to the occlusion sensitivity experiment proposed in [205] is used to identify the face regions from which a CNN extracts the most discriminative features. In the proposed method, the identified face regions are covered during training to force a CNN to extract discriminative features from the non-occluded face regions with the goal of reducing the model's reliance on the identified face regions. The CNN models trained using this approach have demonstrated noticeable performance improvement on face images presenting real-world facial occlusions in the AR face database [206].

The combination of softmax loss and a metric learning loss (e.g. contrastive loss or triplet loss) has provided positive results when training CNN-based face recognition models, either by (i) jointly training with softmax loss and a metric learning loss [159]; or (ii) by first training with softmax loss and then fine-tuning the CNN model with a metric learning loss [11, 164, 165]. In this work, the latter approach is adopted and the triplet loss is used to optimise bottleneck features. The goal of the triplet loss, briefly covered in Section 3.2.5, is to separate positive scores (obtained when comparing pairs of faces belonging to the same subject) from negative scores (obtained when comparing pairs of faces belonging to different subjects) by a minimum margin. In this work, it is argued that training with this loss function can lead to undesired results. Thus, a novel formulation of the triplet loss function is proposed to alleviate this issue by also minimising the standard deviation of both positive and negative scores. Using the Labeled Faces in the Wild (LFW) benchmark, it is shown that the CNN models trained with the proposed loss function consistently outperform those trained with the standard triplet loss function.

The remainder of this chapter is organised as follows. Section 5.2 provides a review of the related work, with a focus on face recognition with occlusion and triplet loss. Section 5.3 explains the method of improving recognition of partially occluded faces and the novel loss function. The experimental results are described in Section 5.4, and the conclusions are presented in Section 5.5.

5.2 Related Work

Recognition of faces with occlusions has been typically handled using two different types of methods, namely, (i) methods that extract local features from the non-occluded regions and (ii) methods that attempt to reconstruct the occluded regions.

In the first type of method, occluded regions are detected first and discarded from the set of local regions used to represent a face. For example, Gabor wavelet features, PCA and SVM were used in [207] to detect occluded regions and LBP descriptors were used to match non-occluded regions. In [208], eigen decomposition was used to generate a reformed image which was subtracted from the original occluded image to locate the occluded regions. Gabor wavelet features and PCA were used to extract features from the non-occluded regions. The method in [209] proposed the extraction of histograms of Gabor-LBP features from the entire image followed by SIFT keypoint matching to select which subregions should be taken into consideration.

Among the methods that attempt to reconstruct occluded regions, the sparse representation-based classification (SRC) proposed in [83] has received a lot of attention. This method attempts to represent an occluded test image by a linear combination of training images and an error term that accounts for the occluded region. More details about SRC and other methods derived from it are given in Section 3.2.2. The drawback of SRC-based methods is that the reconstruction can only be achieved for images of classes that appear in the training set.

Another method that has gained popularity in image reconstruction tasks such as image denoising and image inpainting is the denoising autoencoder [210, 211]. The idea is to train a model to learn a mapping between corrupted and clean images. Several approaches have used this idea to reconstruct occluded face images. For example, a stacked sparse denoising autoencoder [211] with two channels was proposed in [212] to discard noise activations in the encoder network and achieve better image reconstructions. Another related method was proposed in [213]. They used a novel mapping-autoencoder for occlusion detection and an iterative stacked denoising autoencoder for image reconstruction. More recently, [214] proposed to use LSTM autoencoders with two channels to reconstruct faces in the wild. In this method, one autoencoder channel reconstructs the image and the other detects an occlusion mask that is used to replace the occluded region in the original image with the reconstructed pixels. The quality of the final output was further enhanced by introducing an adversarial discriminator.

As discussed in Section 3.2.5, the triplet loss has become one of the most popular training objectives for face verification [137, 11, 215, 164, 165]. Google's FaceNet [137] was the first CNN-based method to use a triplet loss for face recognition. The authors of [137] proposed a novel online triplet sampling strategy that achieves faster convergence by selecting triplets of increasing difficulty during training. The triplet loss has been subsequently used to fine-tune CNNs pre-trained with softmax loss with good results [11, 215, 164, 165]. The triplet loss has also been used in other image similarity tasks such as ranking images [216, 217, 218] and learning local image descriptors [219, 220].

5.3 Proposed Methods

The CNN architecture used in this work is the same as the one proposed in [10], which has demonstrated the ability to achieve high accuracy on the LFW benchmark while maintaining low computational complexity. This CNN architecture is similar to that used in [147] but comprises only ten convolutional layers and one fully-connected layer. The input to this CNN is a greyscale 100×100 image aligned using a simple 2D affine transformation. More details about this CNN architecture can be found in [10].

As a first training stage, the proposed method adopts the approach of training a classifier wherein the CNN produces a vector of scores $\boldsymbol{s}$ for each class j, which is passed to a softmax function to calculate the probability p of the correct class y:

$$p = \frac{e^{s_y}}{\sum_j e^{s_j}} \tag{5.1}$$

The total loss of the CNN is defined as the average cross-entropy loss for each training sample i:

$$L = -\sum_i^n \log p_i \tag{5.2}$$

where n is the number of samples in a batch of training samples.

In order to use the trained CNN classification model to compare face images that are not present in the training set, the classification layer (i.e. the layer producing the scores $\boldsymbol{s}$) is discarded and the features from the previous layer are used as bottleneck features. These bottleneck features can directly be used as the feature vector representing a face or can be further optimised as described in Section 5.3.2. In this work, the cosine similarity is adopted to compare pairs of feature vectors to get a similarity score that indicates the likelihood of two face images belonging to the same identity.

Such a CNN classification model is trained using the CASIA-WebFace database [10]. This database contains 494,414 face images of 10,575 different celebrities gathered from the Internet. 10% of the images are randomly selected to be used as validation images and the rest are used as training images. This CNN model is considered as the baseline for performance comparison in this work and referred to it henceforth as model A.

5.3.1 Occlusions Maps

As shown in [205], it is possible to use visualisation techniques to gain insight into the behaviour of CNN models after they have been trained. To solve the facial occlusion challenge, the face regions a CNN model relies on the most are first identified, as the goal is to avoid this reliance. Using a classification model, one way of visualising these regions is by observing how a correct class score fluctuates when different face regions are occluded. A similar type of occlusion sensitivity experiment has been conducted in [205] in the context of object recognition. In this work, by occluding a face image for which a CNN model predicts the correct class, a *binary occlusion map* $\boldsymbol{O}^{\boldsymbol{I}}$ is generated to indicate whether placing an occluder at a particular spatial location in the image $\boldsymbol{I}$ would cause the model to predict an incorrect class. More formally, a binary occlusion map $\boldsymbol{O}^{\boldsymbol{I}}$ is defined as follows:

$$O^{\boldsymbol{I}}_{i,j} = \begin{cases} 0, & \text{if } \hat{y}_{i,j} = y \\ 1, & \text{otherwise} \end{cases} \tag{5.3}$$

where $\hat{y}_{i,j}$ is the predicted class when the centre of an occluder is placed at the location (i, j) of the image $\boldsymbol{I}$ and y is the correct class for the image $\boldsymbol{I}$.

By using aligned face images, a generic *occlusion map* $\boldsymbol{O}$ can be constructed by simply averaging the binary occlusion maps of a set of face images. Each value of an occlusion map $O_{i,j}$ corresponds to the classification error incurred by a model when an occluder is placed at the location (i, j) in all the images used to generate $\boldsymbol{O}$. For convenience, face regions that present high classification error are referred to henceforth as *high effect regions* (as these are the regions in which the model relies on the most). By contrast, face regions that present low classification error are referred to henceforth as *low effect regions*. These high and low effect regions correspond to the bright and dark areas in the occlusion maps shown in Figure 5.1 respectively.

Considering the 100×100 face images used as input to the proposed model, experiments with occluders of three different sizes are performed. In particular, the following occluders are used: (i) a square occluder of 20×20 pixels that can cover small regions such as one eye, the nose or the mouth as shown in Figure 5.1a; (ii) a rectangular occluder of 20×40 pixels that can cover wider regions such as both eyes simultaneously as shown in Figure 5.1d; and (iii) a larger square occluder of 40×40 pixels that can cover several face regions simultaneously as shown in Figure 5.1g. The occlusion maps generated with the 20×20, 20×40 and 40×40 occluders are denoted by $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$ and $\boldsymbol{O}_{40\times40}$ respectively. Figures 5.1b, 5.1e and 5.1h show an example of these occlusion maps generated with model A using 1,000 images from the validation set.

According to Figures 5.1c, 5.1f and 5.1i, the central part of the face is one of the highest effect regions. This might be due to the presence of non-frontal face images in the training set. Since the

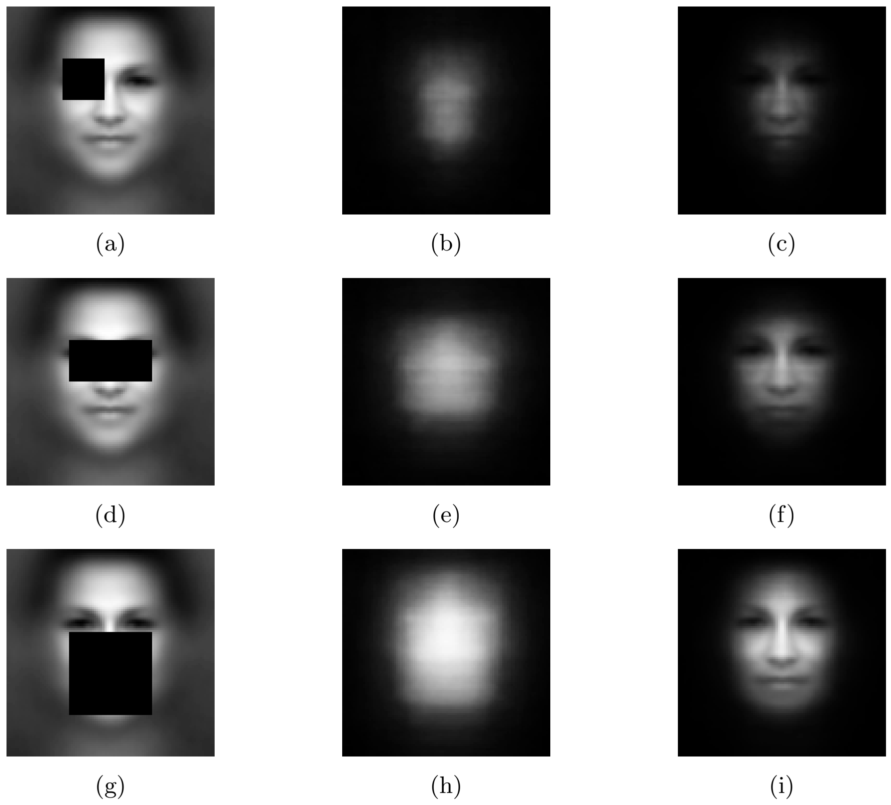
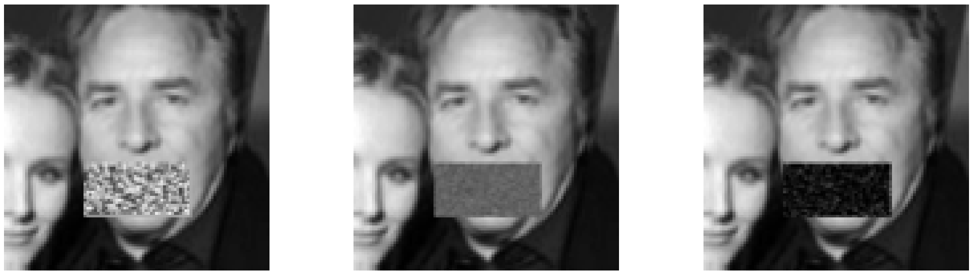


Figure 5.1: (a), (d), (g) Mean image occluded at a random location with an occluder of 20×20, 20×40, and 40×40 respectively. (b), (e), (h) Occlusion maps $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$, and $\boldsymbol{O}_{40\times40}$ generated using model A and the corresponding occluders. The pixel intensity of the occlusion maps represents the classification error rate when placing the occluder at each location. (c), (f), (i) Masked mean image using the occlusion maps $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$, and $\boldsymbol{O}_{40\times40}$ respectively.

central part of the face is typically visible in both frontal and non-frontal face images, the model learns more discriminative features from this area compared to the outer parts of the face, which might not be visible in non-frontal face images. Simply put, the model is trained with fewer face images in which the outer parts of the face are visible, therefore, it relies more heavily on the central part of the face. This behaviour can be reversed by training with more face images that present occlusions located in high effect regions (central part of the face), as this will force the model to learn more discriminative features from low effect regions (outer parts of the face).

One way of achieving this is by augmenting the training set with face images that present occlusions located at random locations. To do this, occluded training images can be generated during training by overlaying the original training images with a randomly located occluder. However, to favour occlusions in high effect regions, the training set is augmented with face images that present occlusions located in high effect regions more frequently than in low effect regions. For this reason,



(a) (b) (c)

Figure 5.2: Example occluders used during training with different intensities, noise types and noise levels. (a) Salt-and-pepper noise. (b) Speckle noise. (c) Gaussian noise.

the location of the occluder is sampled from a probability distribution $\boldsymbol{P}$ generated by applying the softmax function with the temperature parameter t to an occlusion map $\boldsymbol{O}$:

$$P_{i,j} = \frac{e^{\frac{O_{i,j}}{t}}}{\sum_{n,m} e^{\frac{O_{n,m}}{t}}} \tag{5.4}$$

With high temperatures, all locations have the same probability. With low temperatures, locations in high effect regions are assigned a higher probability.

As shown in Figure 5.2, occluders of random intensities (or random colours if the model input were colour images) that present different types of random noise (salt-and-pepper, speckle and Gaussian noise) are used. This is important because if the face is always covered by the same type of occluder, the CNN would only learn features that are robust against that particular type of occlusion. For example, if a black patch is always used to occlude faces during training, the CNN model would perform well when the face is occluded by a black patch, but not when it is occluded by a patch of a different intensity.

This training procedure produces two desired outcomes, namely, (i) the training set is augmented with variations not present in the original data, and (ii) the occluder has a regulariser effect, helping the CNN to learn features from all face regions equally. Both of these increase the generalisation capability of the model and prevent overfitting. Section 5.4 provides experimental results, with both occluded and non-occluded face images, to validate these claims.

5.3.2 Batch Triplet Loss

In order to make the bottleneck features generalise better to classes not present in the training set, model A is fine-tuned using a triplet loss function. This training objective is also used in other similar works [11, 215, 164, 165]. However, in this work, the bottleneck features are fine-tuned directly instead of learning a linear projection from them. It could be argued that the CNN model could be trained from scratch using a triplet loss function, as proposed in [137]. But, according to

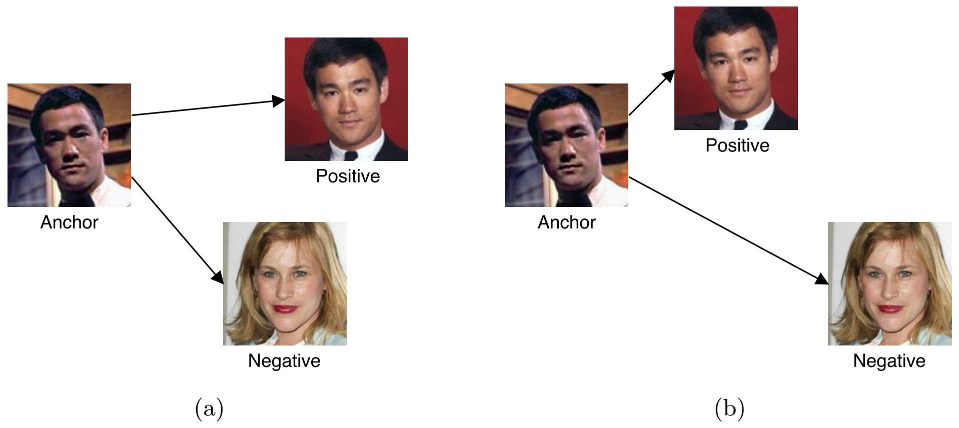


Figure 5.3: Example triplet from the CASIA-WebFace dataset. (a) Before triplet training. (b) After triplet training.

the experiments performed in this work, training with softmax loss offers faster convergence than training with a triplet loss when a reasonable number of samples per class are available and the number of classes is not very large.

To form a triplet, an anchor image, a positive image and a negative image are needed. The anchor and the positive images belong to the same class and the negative image belongs to a different class. Denoting the output vector of the CNN model as $\boldsymbol{z}$ (these are the bottleneck features in the proposed model), the output features for a particular triplet i can be represented as $(\boldsymbol{z}_i^a, \boldsymbol{z}_i^p, \boldsymbol{z}_i^n)$, denoting the output features for the anchor, positive and negative images respectively. The goal of a triplet loss function is to make the distance between $\boldsymbol{z}_i^a$ and $\boldsymbol{z}_i^n$ (i.e. images from different classes) larger than the distance between $\boldsymbol{z}_i^a$ and $\boldsymbol{z}_i^p$ (i.e. images from the same class) by at least a minimum margin α. Figure 5.3 shows a visual representation of a triplet before and after training. In this work, the following is considered as the standard triplet loss function:

$$L = \sum_i^N \max\left(0, \|\boldsymbol{z}_i^a - \boldsymbol{z}_i^p\|_2^2 - \|\boldsymbol{z}_i^a - \boldsymbol{z}_i^n\|_2^2 + \alpha\right) \tag{5.5}$$

Alternative versions of the standard triplet loss can be defined with distance metrics other than the squared Euclidean distance. For example, the dot product is used as the similarity measure in [164]. A more general form can be written as:

$$L = \sum_i^N \max\left(0, d(\boldsymbol{z}_i^a, \boldsymbol{z}_i^p) - d(\boldsymbol{z}_i^a, \boldsymbol{z}_i^n) + \alpha\right) \tag{5.6}$$

where $d(\boldsymbol{a}, \boldsymbol{b})$ is any function that gives a score indicating distance between two feature vectors. As seen in Equation 5.6, only triplets that violate the margin condition $d(\boldsymbol{z}_i^a, \boldsymbol{z}_i^p) + \alpha > d(\boldsymbol{z}_i^a, \boldsymbol{z}_i^n)$

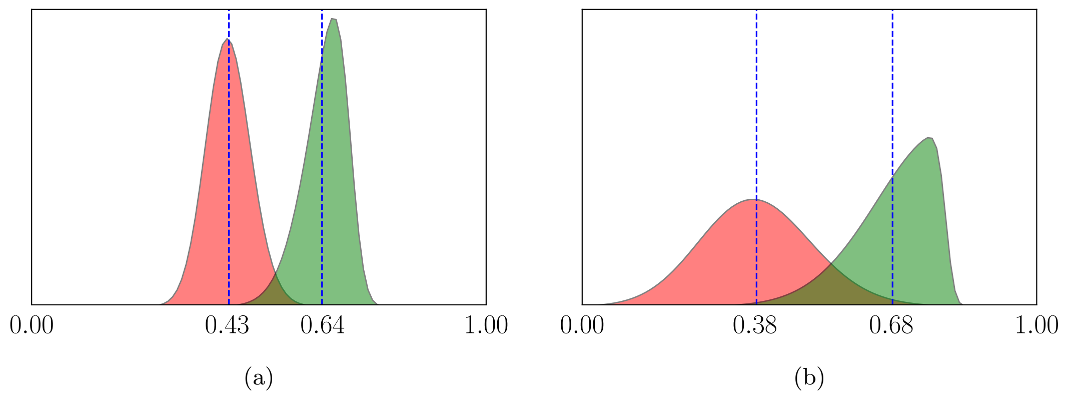


Figure 5.4: (a) Distribution of positive and negative scores after training a CNN classification model. (b) Distribution of positive and negative scores after fine-tuning the same CNN model with the standard triplet loss. Observe how even though the triplet training has been able to further separate the mean values of the two distributions, there is more overlapping between them, causing more false positives and/or false negatives

produce a loss greater than zero and therefore contribute to the model's convergence. To increase the training efficiency, the online triplet sampling strategy proposed in [137] is adopted to select such triplets and only use them during training. Taking this into consideration, Equation 5.6 can be rewritten as:

$$L = \mu_{ap} - \mu_{an} + \alpha \tag{5.7}$$

where μ_{ap} and μ_{an} are the mean values of the distribution of positive and negative scores respectively.

From Equation 5.7 it can be seen that the loss becomes zero whenever μ_{an} is equal to μ_{ap} plus the margin α. In other words, the triplet loss tries to separate the mean values of the distribution of positive scores μ_{ap} from the mean value of the distribution of negative scores μ_{an} by a minimum margin α.

A problem with the standard triplet loss is that, in general, separating the mean values of the two score distributions does not ensure that the model performs well in a verification task. Figure 5.4 shows how a CNN model that has been fine-tuned with the standard triplet loss is able to further separate the mean values of the two score distributions but does not produce a better accuracy. This is because there might be more overlapping between the two distributions, causing more false positives and/or false negatives. A solution to this problem is to also minimise the standard deviation of each score distribution. The proposed loss function is inspired by the concept of *decidability*, proposed in [221] as a way of measuring the achievable accuracy of a verification system regardless of the selected threshold or operating point. A possible measure of decidability is defined as follows [221]:

$$d = \frac{|\mu_{ap} - \mu_{an}|}{\sqrt{\frac{1}{2}\left(\sigma_{ap}^2 + \sigma_{an}^2\right)}} \tag{5.8}$$

where σ_{ap}^2 and σ_{an}^2 are the variances of the distributions of positive and negative scores respectively.

Equation 5.8 implies that a higher decidability d is achieved by increasing the difference between the mean values of the two score distributions while decreasing both of their variances. Although it would be possible to use the inverse of Equation 5.8 as the training objective, in practice, using the margin parameter α leads to a better separation between the two score score distributions. For this reason, the proposed loss function is constructed by adding a new term to Equation 5.7 that accounts for the variance in the two score distributions:

$$L = (1 - \beta)\left(\mu_{ap} - \mu_{an} + \alpha\right) + \beta\left(\sigma_{ap}^2 + \sigma_{an}^2\right) \tag{5.9}$$

where β is a parameter that balances the contribution of the two terms. In particular, at $\beta = 1$, the term that accounts for the difference between the mean values of each score distribution vanishes and only the term that accounts for the variances of the score distributions has an effect. The opposite happens when $\beta = 0$.

An advantage of adding this new term to the triplet loss function is that even if a triplet does not violate the margin condition, the loss will usually be greater than zero since the term that accounts for the variances of the score distributions is non-zero. Even though this means that adopting an online triplet sampling strategy is not strictly needed, a faster convergence was observed in the experiments when using it. Concurrent to this work, a similar loss function has been proposed in [220] to learn local image descriptors. However, [220] does not make use of online triplet sampling.

Note that the loss function in Equation 5.9 cannot be expressed as the average loss for each training image since the variances need to be computed with more than one sample. Ideally, large enough batches of images need to be used during training so that the variance estimation is more accurate. For this reason, this work refers to this form of triplet loss as *batch triplet loss*. Section 5.4.2, shows the improved accuracy when using the proposed batch triplet loss function compared to the standard triplet loss function.

5.4 Experiments

This section provides experimental results for the two contributions presented in this work. In Section 5.4.1, different CNN models trained with occluded training images as described in Section 5.3.1 are evaluated. The CASIA-WebFace database [10] is used to evaluate the performance on face images that present artificial occlusions and the AR face database [206] is used to evaluate the performance on face images that present real-world occlusions. Section 5.4.2 shows the experimental results on the LFW [88] benchmark using the CNN models evaluated in Section 5.4.1 and their fine-tuned

Model	$\boldsymbol{O}_{20\times20}$	$\boldsymbol{O}_{20\times40}$	$\boldsymbol{O}_{40\times40}$
A	$92.9\% \pm 10.99$	$86.18\% \pm 18.51$	$76.19\% \pm 27.89$
$\mathrm{A}_{20\times20}$	$\mathbf{97.69\% \pm 2.62}$	$\mathbf{95.1\% \pm 5.64}$	$\mathbf{88.9\% \pm 14.13}$
$\mathrm{A}_{20\times20\mathrm{R}}$	$97.12\% \pm 3.55$	$93.98\% \pm 7.39$	$86.93\% \pm 16.98$
$\mathrm{A}_{20\times40}$	$\mathbf{97.75\% \pm 2.42}$	$\mathbf{95.85\% \pm 4.03}$	$\mathbf{90.64\% \pm 10.88}$
$\mathrm{A}_{20\times40\mathrm{R}}$	$97.62\% \pm 2.9$	$95.45\% \pm 5.12$	$89.54\% \pm 13.29$
$\mathrm{A}_{40\times40}$	$\mathbf{98.37\% \pm 1.7}$	$\mathbf{96.8\% \pm 3.16}$	$\mathbf{93.47\% \pm 6.94}$
$\mathrm{A}_{40\times40\mathrm{R}}$	$98.31\% \pm 2.29$	$96.52\% \pm 4.14$	$92.61\% \pm 9.13$

Table 5.1: Mean classification accuracy and standard deviation of each occlusion map $\boldsymbol{O}$ generated by different CNN models.

versions using the standard triplet loss and the proposed batch triplet loss.

5.4.1 Performance on Occluded Faces

Section 5.3.1 described a training procedure for increasing the CNN model classification accuracy on occluded faces by using a probability distribution $\boldsymbol{P}$ to augment the training set with occluded training images. This section will start by comparing the performance of two different training schemes. The first training scheme comprises fine-tuning the baseline model A, described in Section 5.3, with occluded training images generated by sampling the occluder locations from a probability distribution $\boldsymbol{P}$. The probability distribution $\boldsymbol{P}$ is obtained by applying Equation 5.4 to an occlusion map $\boldsymbol{O}$ of a particular size. Each occlusion map $\boldsymbol{O}$ was generated with model A using a subset of 1,000 images from the CASIA-WebFace validation set, as described in Section 5.3.1. By contrast, the second training scheme comprises fine-tuning model A with occluded training images generated by sampling the occluder locations from a standard normal distribution. The goal of training with these two training schemes is to assess the benefits of training CNN models with images overlaid by strategically located occluders as opposed to randomly located occluders.

Several CNN models are trained in this manner, one for each of the occluder sizes shown in Figure 5.1. The temperature value t in Equation 5.4 was empirically set to 0.25, 0.4 and 0.6 with $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$ and $\boldsymbol{O}_{40\times40}$ respectively. The size of the occluder used to generate the occluded training images is added to the name of each fine-tuned model. Additionally, if the model was trained following the second training scheme, an R is added to the model name. For example, model A fine-tuned with occluded training images overlaid by an occluder of 20×20 pixels becomes $\mathrm{A}_{20\times20}$ if the locations of the occluder are sampled from $\boldsymbol{P}$ (first training scheme) and $\mathrm{A}_{20\times20\mathrm{R}}$ if the locations of the occluder are sampled from a standard normal distribution (second training scheme).

The accuracy of these fine-tuned CNN models is evaluated by generating occlusion maps $\boldsymbol{O}$ with them (one for each of the occluder sizes). Since an occlusion map indicates the classification error incurred by a model at each spatial location, the mean classification accuracy can be easily

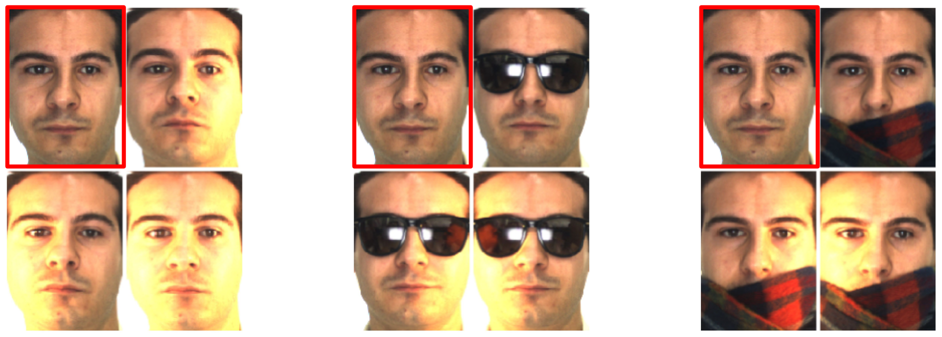
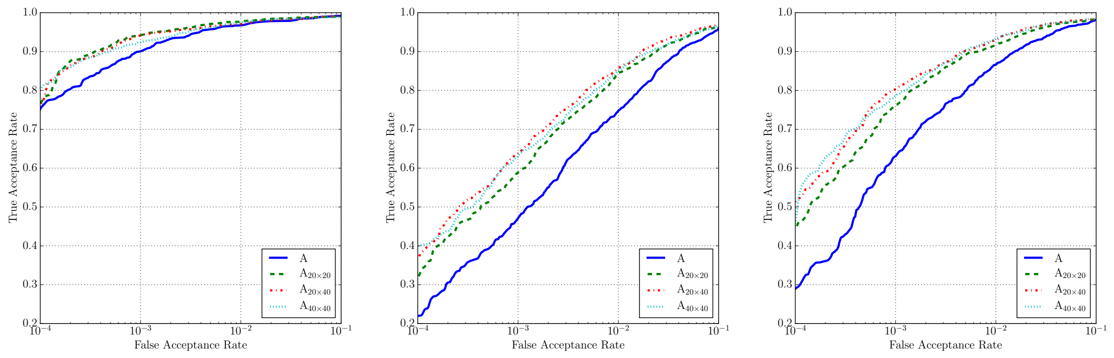


Figure 5.5: Example images from the AR database. In each subfigure, the highlighted image on the top left is the reference image (target image) used to compare against the other three images (query images). (a) Non-occluded. (b) Wearing sunglasses. (c) Wearing scarf.

calculated as $1-\sum_{i,j} O_{i,j}$. Table 5.1 shows the mean classification accuracy and standard deviation for each occlusion map generated with each fine-tuned model. For each model in Table 5.1, the three occlusion maps $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$ and $\boldsymbol{O}_{40\times40}$ are generated using a subset of 1,000 images from the CASIA-WebFace validation set in such a way that all the selected images can be correctly classified by the model if no occluder is used. For example, to generate $\boldsymbol{O}_{20\times20}$, $\boldsymbol{O}_{20\times40}$ and $\boldsymbol{O}_{40\times40}$ with model $\text{A}_{20\times20}$, a subset of 1,000 images that were classified correctly by model $\text{A}_{20\times20}$ was used. To avoid any bias in the results, a different subset of 1,000 images was selected to generate the occlusion maps used to compute the results shown in Table 5.1 and to generate the probability distribution $\boldsymbol{P}$ used when training each model. In other words, it was avoided testing the models using the same images that were (indirectly) incorporated in the training stage by the use of $\boldsymbol{P}$. Observe that not only all the models fine-tuned with occluded training images achieve a higher classification accuracy than model A but their standard deviations are considerably smaller. This indicates that the performance of the fine-tuned models is much less affected by the location of the occluder, i.e. the models are able to extract discriminant features from all the face regions more equally, regardless of the location of the occluder. Moreover, the results in Table 5.1 show the better performance of the models trained with occluded training images overlaid by strategically located occluders compared to those trained with occluded training images overlaid by randomly located occluders.

Since the goal is to improve the accuracy when dealing with real-world occlusions, the performance of the CNN models has been further evaluated on the AR face database [206]. The AR face database contains 4,000 face images of 126 different subjects with different facial expressions, illumination conditions and occlusions. Out of these, only faces with different illumination conditions and occlusions are used. The different illumination conditions correspond to face images with a light on the left side, right side or both. The occluded face images consist of people wearing either sunglasses or a scarf. Three different evaluations are carried out. In each evaluation, non-occluded



(a) (b) (c)

Figure 5.6: AR database ROC curves (a) Non-occluded. (b) Wearing sunglasses. (c) Wearing scarf.

faces are compared against (i) other non-occluded faces, (ii) faces occluded by a pair of sunglasses, and (iii) faces occluded by a scarf. Figure 5.5 shows examples of each type of image used in the three evaluations.

As shown by the resulting ROC curves in Figures 5.6a to 5.6c, the performance of the models trained with occluded training images consistently outperform the baseline model A, particularly at low FAR. Note that the performance does not seem to be greatly affected by the occluder size. The ROC curve for the evaluation of non-occluded faces (Figure 5.6a) shows that model $\mathrm{A}_{40\times40}$ performs slightly worse than models $\mathrm{A}_{20\times20}$ and $\mathrm{A}_{20\times40}$, perhaps because the large occluder used during training makes the model rely on fewer features. As a consequence, the model performs worse when presented with non-occluded faces in which all the face regions are visible and contain useful features. In contrast, model $\mathrm{A}_{20\times20}$ performs worse than the other two when presented with faces occluded by a pair of sunglasses (Figure 5.6b) or a scarf (Figure 5.6c). This might be because the occluder used during training is too small to simulate these types of occlusions.

Observe that, even though model $\mathrm{A}_{40\times40}$ achieved the best classification accuracy when evaluated on face images that present artificial occlusions (Table 5.1), the results on the AR face database differ because the evaluation involves comparing pairs of occluded and non-occluded face images instead of only classifying occluded face images. For this reason, the models need to perform well not only when presented with occluded face images but also with non-occluded face images. It seems that using a medium-sized occluder like the one used to train model $\mathrm{A}_{20\times40}$ offers the best performance when taking into account the three different evaluations, as it avoids the problems encountered with small occluders (not being able to simulate large occlusions like sunglasses and scarves) and large occluders (worse performance when presented with non-occluded faces). Note that these experiments were not repeated with the models trained using the second training scheme described earlier, as their performance was already shown to be inferior.

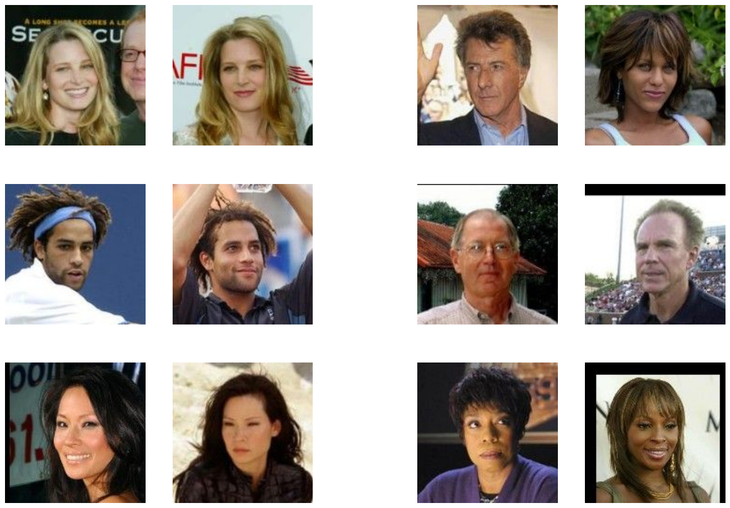

(a) (b)

Figure 5.7: Example pairs from the LFW benchmark. (a) Positive pairs. (b) Negative pairs.

5.4.2 Performance on the LFW benchmark

In this subsection, another approach to training is adopted by fine-tuning model A using the standard triplet loss and the batch triplet loss described in Section 5.3.2. This is done by discarding the classification layer, normalising the features of the previous layer (bottleneck features) using the L_2-norm and fine-tuning all the CNN layers with one of the two loss functions. The CNN model fine-tuned with the standard triplet loss function is referred to henceforth as model B, and the CNN model fine-tuned with the batch triplet loss function is referred to henceforth as model C. The parameter α is set to 0.5 when using any of these training objectives and the parameter β is set to 0.7 when training with the batch triplet loss function. The values for both α and β were obtained empirically.

Additionally, these CNN models were also trained with occluded training images. Similar to the notation followed in Section 5.4.1, the size of the occluder used during training is appended to the model name. In this case, these models were trained by fine-tuning a model that had already been trained with occluded training images instead of fine-tuning model A. The locations of the occluders were sampled from the same probability distributions $\boldsymbol{P}$ that were used in Section 5.4.1. For example, to train model $\mathrm{B}_{20\times40}$, model $\mathrm{A}_{20\times40}$ (and not model A) was fine-tuned with occluded training images overlaid by an occluder of 20×40 placed at locations sampled from the same probability distribution $\boldsymbol{P}$ that was used to train $\mathrm{A}_{20\times40}$.

The models are evaluated on the LFW dataset following the *unrestricted, labeled outside data* protocol [222] (i.e. the protocol that allows training with data that is not part of the LFW dataset).

Model	Accuracy
A	97.33% ± 0.71
B	97.73% ± 0.76
C	**98.12% ± 0.65**
$A_{20\times20}$	97.4% ± 0.71
$B_{20\times20}$	97.85% ± 0.69
$C_{20\times20}$	**98.35% ± 0.73**
$A_{20\times40}$	97.68% ± 0.83
$B_{20\times40}$	97.79% ± 0.82
$C_{20\times40}$	**98.42% ± 0.68**
$A_{40\times40}$	97.18% ± 0.63
$B_{40\times40}$	97.5% ± 0.57
$C_{40\times40}$	**98.16% ± 0.64**

Table 5.2: Mean classification accuracy and standard deviation of different CNN models evaluated following the LFW unrestricted, labeled outside data protocol.

The LFW protocol divides the test set in ten splits. The classification accuracy on each test split is calculated by counting the number of correct positive and negative pairs given a certain threshold. For each test split, the threshold that gives the highest amount of correct classifications in the other nine splits was selected. The final reported value is the mean classification accuracy and the standard deviation calculated from the ten test splits. Note that most of the face images in the LFW dataset are not occluded. Therefore, a performance improvement as large as that seen in the experiments with occluded faces in Section 5.4.1 is not expected. Figure 5.7 shows examples of positive and negative pairs of face images from the LFW benchmark.

The evaluation presented here does not compare against other similar works since the results obtained when evaluating CNN-based methods is highly dependent on training data and CNN architecture, which are not the focus of this work. Instead, the results in Table 5.2 show the improvements made by the contributions presented in this chapter with respect to the baseline model A. As shown in Table 5.2, all the CNN models fine-tuned with the batch triplet loss outperform the CNN models trained with the standard triplet loss, validating the usefulness of the proposed approach. Moreover, consistent with the results shown in Figure 5.6, the CNN models trained with the 20×40 occluder are the best performers.

5.5 Conclusions

This work has investigated which parts of the human face have the highest impact on face recognition accuracy. The proposed occlusion maps are a good way of visualising these regions and, at the same time, provide useful information about a classification model's performance on faces that present artificial occlusions. According to the experimental results, even a state-of-the-art CNN-based face recognition model fails to maintain its high performance when these face regions are occluded (e.g. by a pair of sunglasses or a scarf). It has been demonstrated how these occlusion maps can be used during the training procedure to augment the training set with face images that present artificial occlusions. These artificial occlusions are strategically positioned in locations where the performance of a CNN model trained in a conventional way is most sensitive. By training with these augmented training sets, CNN models that are more robust to face occlusions are obtained. As shown in the experimental results, the proposed method has shown consistent performance improvement on face images that present artificial or real-world occlusions and on face images that do not present any occlusions.

Additionally, the problem of learning features for a verification task using metric learning has been revisited. The widely used triplet loss has been extented by adding a new term that minimises the standard deviation of the distributions of positive and negative scores. In the experiments on the LFW benchmark, the proposed batch triplet loss has consistently achieved better results than the standard triplet loss. Finally, the experimental results have confirmed that the best CNN models result from a combination of the two proposed approaches, regardless of whether the face images are occluded or not.

Since the development of the methods proposed in this chapter, several models based on generative adversarial networks have been proposed for facial attribute editing [170, 172, 173, 175, 174]. These methods could be used to generate photo-realistic occlusions instead of the simple artificial occluders used in this work. Regarding the second contribution, even though the recently proposed methods based on softmax loss with margin [151, 154, 155, 156] have emerged as the leading approach for training CNN-based face recognition models, it has been shown that their performance can still benefit from metric learning methods such as the triplet loss [156].

The next chapter follows-up on the idea of data augmentation for face recognition. However, instead of generating simple modifications of existing face images as proposed here, a generative model is used to generate photo-realistic face images of both existing subjects in a dataset and new subjects.

Chapter 6

Generating Photo-Realistic Training Data to Improve Face Recognition Accuracy

This chapter investigates the feasibility of using synthetic data to augment face datasets. In particular, a novel generative adversarial network (GAN) that can disentangle identity-related attributes from non-identity-related attributes is proposed. This is done by training an embedding network that maps discrete identity labels to an identity latent space that follows a simple prior distribution, and training a GAN conditioned on samples from that distribution. The proposed GAN can be used to augment face datasets by generating both synthetic images of subjects in the training set and synthetic images of new subjects not in the training set. By using recent advances in GAN training, it is shown that the synthetic images generated by the proposed model are photo-realistic, and that training with augmented datasets can indeed increase the accuracy of face recognition models as compared to models trained with real images alone.

This work was done during the last year of this PhD project (2018) and will serve as the basis for another academic publication[1].

6.1 Introduction

Image synthesis is a widely studied topic in computer vision. In particular, face image synthesis has gained a lot of attention because of its diverse practical applications. These include facial image editing [169, 223, 170, 171, 185, 184, 174, 176, 177], face de-identification [224, 225, 226, 227],

[1]An online preprint is available as *Generating Photo-Realistic Training Data to Improve Face Recognition Accuracy*, arXiv preprint arXiv:1811.00112, 2018.

face recognition (e.g. data augmentation [228, 229, 230, 231, 232, 233, 234] and face frontalisation [235, 236, 53, 237, 238, 183, 182]) and artistic applications (e.g. video games and advertisements).

This work focuses on the applicability of face image synthesis for data augmentation. It is widely known that training data is one of the most important factors that affect the accuracy of deep learning models. The datasets used for training need to be large and contain sufficient variation to allow the resulting models to learn features that generalise well to unseen samples. In the case of face recognition, the datasets must contain many different subjects, as well as many different images per subject. The first requirement enables a model to learn inter-class discriminative features that can generalise to subjects not in the training set. The second requirement enables a model to learn features that are robust to intra-class variations. Even though there are several public large-scale datasets [9, 10, 11, 12, 13, 14, 15] that can be used to train CNN-based face recognition models, these datasets are nowhere near the size or quality of commercial datasets. For example, the largest publicly available dataset contains about 10M images of 100K different subjects [12], whereas Google's FaceNet [137] was trained with a private dataset containing between 100M and 200M face images of about 8M different subjects. Another issue is the presence of long-tail distributions in some publicly available datasets, i.e. datasets in which there are many subjects with very few images. Such unbalanced datasets can make the training process difficult and result in models that achieve lower accuracy than those trained with smaller but balanced datasets [232, 145]. In addition, some publicly available datasets (e.g. [12]) contain many mislabelled samples that can decrease face recognition accuracy if not discarded from the training set. Since collecting large-scale, good quality face datasets is a very expensive and labour-intensive task, this work proposes a method for generating photo-realistic face images that can be used to effectively increase the depth (number of images per subject) and width (number of subjects) of existing face datasets.

An approach that has recently gained popularity for augmenting face datasets is the use of 3D morphable models [239]. In this approach, new faces of existing subjects can be synthesized by fitting a 3D morphable model to existing images and modifying a variety of parameters to generate new poses and expressions [228, 231, 232, 234]. It is also possible to generate images with other variations using this approach. For example, [234] incorporated a reflectance model to generate images under different lighting conditions; and [233] randomly sampled 3D face shapes and colours to generate faces of new subjects. The main drawback of methods based on 3D morphable models is that the generated images often look unnatural and lack the level of detail found in real images. Another recent approach based on blending small triangular regions from different training images was proposed in [229]. Although this method seemed to produce photo-realistic faces, the authors limited their work to frontal face images. In contrast, the proposed approach makes use of generative adversarial networks (GANs) [49], which have recently been shown to produce photo-realistic in-the-wild images often indistinguishable from real images [240]. Another advantage of using GANs is that they are end-to-end trainable models that do not require any domain-specific processing, as opposed to methods based on 3D modelling or face triangulation.

Many methods based on GANs have been proposed for manipulating attributes of existing face images, including age [223, 185, 184], facial expressions [223, 174, 179, 180], and other attributes such as hairstyle, glasses, makeup, facial hair, skin colour or gender [169, 170, 171, 174, 176, 177, 172, 173, 178]. While these methods can be used to increase the depth of a dataset, it remains unclear how to increase the width of a dataset, i.e. how to generate faces of new subjects. The proposed GAN is able to generate faces from a latent representation $\boldsymbol{z}$ that has two gaussian distributed components $\boldsymbol{z}_{id}$ and $\boldsymbol{z}_{nid}$ encoding identity-related attributes and non-identity-related attributes respectively. In this way, face images of new subjects can be generated by fixing the identity component $\boldsymbol{z}_{id}$ and varying the non-identity component $\boldsymbol{z}_{nid}$. The method most closely related to this work is the *semantically decomposed GAN* (SD-GAN) proposed in [181]. However, in contrast with SD-GAN, the proposed method also supports the generation of face images of subjects that exist in the training set. In other words, the proposed method can increase both the width and the depth of a given face dataset. Furthermore, the proposed GAN is arguably simpler to implement than SD-GAN and easier to incorporate into other GAN architectures.

To demonstrate the efficacy of the proposed method, several CNN-based face recognition models were trained with different combinations of real and synthetic data. In most cases, the models trained with a combination of real and synthetic data outperformed the models trained with real data alone.

The rest of this chapter is organised as follows. Section 6.2 provides a review of related GAN methods. Section 6.3 explains each part of the proposed GAN and the loss functions used for training. Section 6.4 discusses the experimental results, both in terms of the quality of the synthetic images generated by the proposed GAN and the accuracy achieved by datasets augmented with the synthetic images. Finally, the conclusions are presented in Section 6.5.

6.2 Related Work

The literature on face image synthesis is very extensive. This section focuses on GAN methods related to the method proposed in this work. For a recent survey on different face image synthesis methods (including GANs) see [241].

As explained in Section 2.3.3, GANs generate data by sampling from a probability distribution p_{model} that is trained to match a true data generating distribution p_{data}. This is done by mapping a vector of random latent variables $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ to a sample $G(\boldsymbol{z})$ through a generator network G, where $p_{\boldsymbol{z}}$ is a prior distribution that can be easily sampled (e.g. Gaussian or uniform). The generator is trained to fool a discriminator network D that tries to determine whether a sample is real or generated (i.e. synthetic). Thus, the generator and discriminator are trained with opposing optimisation objectives. While the discriminator is trained to maximise the probability of correctly classifying both real and generated samples, the generator is trained to minimise the probability that the generated samples are classified as such. Formally, the standard GAN optimisation objective can be expressed as follows

$$\min_{G} \max_{D} \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))] \tag{6.1}$$

As training progresses, the discriminator gets better at distinguishing real from generated samples and the generator gets better at producing realistic samples that can fool the discriminator. The training is considered completed when the generator and the discriminator reach an equilibrium, i.e. when the generator and the discriminator stop improving. In practice, since GANs rarely reach an equilibrium, it is common to simply stop the training process whenever there is no noticeable improvement in the visual quality of the generated samples.

The training of GANs is often unstable and can lead to the mode collapse problem (this happens when the generator maps different values of $\boldsymbol{z}$ to the same output sample [242]). Although some works have proposed heuristics that reduce this effect [243, 244], a full understanding of the training dynamics of GANs remains an open research question. Based on the idea that optimising the training objective in Equation 6.1 can be interpreted as minimising the Jensen-Shannon divergence between the true data generating distribution p_{data} and the model distribution p_{model} [49], the authors of [52] proposed a novel training objective for GANs that minimises the Wasserstein distance instead of the Jensen-Shannon divergence. This GAN variation, which was named the Wasserstein GAN (WGAN), was shown to be more stable and to reduce the mode collapse problem [52]. An improved formulation of this approach [245] is considered one of the current state-of-the-art techniques for training GANs.

Many works on GANs have adopted a family of architectures known as *deep convolutional GANs* (DCGANs) [243]. DCGANs follow a set of guidelines that were proposed for stable training and good image quality. More recently, [240] proposed a new methodology for training GANs that consist of progressively growing both the spatial resolution of real and generated images and the number of layers of the discriminator and generator networks (PGGAN). In this manner, the training is very stable at the start since low-resolution images are easier to generate than high-resolution images due to their lower dimensionality and hence diversity. As training progresses, and the resolution of the images is increased, the generator gradually learns to generate images with finer detail. In contrast, standard GANs that are tasked with learning high-resolution images from the outset are typically more unstable. Using the PGGAN approach together with several proposed heuristics, the authors of [240] were able to generate impressive photo-realistic 1024×1024 images with a $5.4\times$ speedup factor with respect to the standard GAN training approach. The proposed GAN uses this approach since it was the state-of-the-art for face image generation at the time this work was done.

Conditional versions of GANs allow the generation of samples with specific attributes. The first conditional GAN was introduced in [246] and consisted of feeding a label $\boldsymbol{y}$ encoding some attribute(s) of the data to both the generator and the discriminator. Using this approach, the input to the generator can be constructed by concatenating the vector of latent variables $\boldsymbol{z}$ with the label $\boldsymbol{y}$. Likewise, the input to the discriminator can be constructed by combining the input image $\boldsymbol{x}$ and the label $\boldsymbol{y}$ (for example, by upscaling $\boldsymbol{y}$ and appending it as an additional channel to $\boldsymbol{x}$). Several applications based on this idea have been proposed, such as facial attribute editing [170, 173], facial expression manipulation [180] and face ageing [185, 184]. An alternative type of conditional GAN called auxiliary classifier GAN (AC-GAN) was proposed in [247]. Instead of feeding the label $\boldsymbol{y}$ to the discriminator, AC-GANs use an auxiliary classifier in the discriminator that is tasked with predicting the label $\boldsymbol{y}$ that has been fed to the generator. AC-GANs have also been successfully used for various applications, including facial attribute editing [178], face frontalisation [183] and multi-domain image-to-image translation [174]. Moreover, the use of an auxiliary classifier to predict $\boldsymbol{y}$ was shown in [248, 244] to improve image quality even when $\boldsymbol{y}$ was not fed to the generator.

In the case of face image synthesis, conditional GANs can generate images of subjects existing in the training set by considering the identity of a subject as an attribute $\boldsymbol{y}$. However, this approach does not allow generation of samples of new subjects. The SD-GAN method proposed in [181] is the only work that has attempted to solve this task. SD-GANs split the vector of latent variables $\boldsymbol{z}$ into two components $\boldsymbol{z}_I$ and $\boldsymbol{z}_O$ encoding identity-related attributes and non-identity-related attributes respectively. SD-GANs are trained with pairs of real images from the same subject and pairs of images generated with the same identity-related attributes $\boldsymbol{z}_I$ but different non-identity-related attributes $\boldsymbol{z}_O$. The discriminator learns to reject pairs of images when either they do not look photo-realistic or when they do not appear to belong to the same subject. Once the training is completed, since the latent variables $\boldsymbol{z}_I$ are forced to encode identity-related attributes, images of new subjects can be generated by feeding the generator with random $\boldsymbol{z}_I$ samples. Although SD-GANs have a similar goal to the proposed GAN, they cannot generate images of subjects in the training set. The proposed GAN can generate images of subjects in the training set and uses a simpler architecture that does not need to be trained with pairs of images.

An approach related to conditional GANs is the InfoGAN model proposed in [249]. The goal of an InfoGAN is to disentangle data attributes in an unsupervised way. This is done by maximising the mutual information [250] between a subset $\boldsymbol{c}$ of the latent variables $\boldsymbol{z}$ and the generated image $G(\boldsymbol{z})$. InfoGANs can be implemented with an auxiliary network in the discriminator that is trained to predict $\boldsymbol{c}$. As shown in [249], this method ensures that the latent variables $\boldsymbol{c}$ encode meaningful data attributes that are not lost during the generation process. The proposed GAN incorporates elements of both conditional GANs and InfoGANs to disentangle identity-related attributes from non-identity-related attributes.

6.3 Proposed Method

In this section, the choice of GAN architecture and type of conditional GAN is first explained, and then the proposed modifications to disentangle identity-related attributes from non-identity-related attributes.

6.3.1 Conditional PGGAN

In this work, the same architecture and training method proposed in the PGGAN work [240] were used and modified as needed (the open-source code released by the authors was used and the default training settings were kept unless otherwise stated). The training starts by generating 4×4 images. The number of layers in the generator and discriminator is then gradually increased from 1 to 11 (each time the resolution is doubled, two convolutional layers are added) until 128×128 images are generated. Higher resolution images are not generated because the network that is used for face recognition in the experiments takes 100×100 images as input. Following [240], instead of using the standard GAN objective shown in Equation 6.1, the WGAN training objective [245] is used:

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))] \tag{6.2}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions (for a full derivation of Equation 6.2 see [52]). When posed as a minimisation problem and adding a gradient penalty that enforces the Lipschitz constraint (the gradient of 1-Lipschitz functions are bounded to 1), the WGAN loss [245] becomes:

$$L_{img} = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))] - \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[D(\boldsymbol{x})] + \lambda\ \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{\tilde{\boldsymbol{x}}}}[\|(\nabla_{\tilde{\boldsymbol{x}}} D(\tilde{\boldsymbol{x}})\|_2 - 1)^2] \tag{6.3}$$

where $p_{\tilde{\boldsymbol{x}}}$ is a distribution of samples interpolated from generated samples $G(\boldsymbol{z})$ and real samples $\boldsymbol{x}$, and λ is a weight controlling the contribution of the gradient penalty to the loss. Following [245], the value of λ is set to 10.

To make this model conditional, the AC-GAN method is used, i.e. the generator is conditioned on identity labels $\boldsymbol{y}$ and an auxiliary network D_c in the discriminator D is trained to predict $\boldsymbol{y}$. Since the identity labels $\boldsymbol{y}$ are categorical, D_c is trained as a classifier with cross-entropy loss:

$$L_c = -\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}, \boldsymbol{y} \sim p_{\boldsymbol{y}}}[\log D_c(G(\boldsymbol{z} \mid \boldsymbol{y}))] - \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D_c(\boldsymbol{x})] \tag{6.4}$$

Note that the generator is also trained to minimise this loss function so that the identity labels $\boldsymbol{y}$ are not ignored during the generation process.

6.3.2 Identity Latent Space

The model described in Section 6.3.1 can generate images of subjects with identities $\boldsymbol{y}$ existing in the training set. However, it is not possible to generate images of subjects with new identities. For this reason, the use of an embedding network E is proposed to map the discrete identity labels $\boldsymbol{y}$ to a vector of latent variables $E(\boldsymbol{y})$. Since the goal is to learn a continuous latent space of identities that can be sampled from, $E(\boldsymbol{y})$ is trained to follow a simple prior distribution $p_{\boldsymbol{z}_{id}}$ such as the Gaussian distribution used in this work. This can be done by using another discriminator $D_{\boldsymbol{z}_{id}}$ that is trained to match the posterior distribution defined by $E(\boldsymbol{y})$ to the Gaussian prior distribution $p_{\boldsymbol{z}_{id}}$ using adversarial training, as proposed in [251]. To incorporate the identity latent space into the conditional PGGAN model from Section 6.3.1, the generator network G is conditioned on the latent representation of the labels, i.e. $G(\boldsymbol{z} \mid E(\boldsymbol{y}))$. A diagram of the proposed GAN is shown in Figure 6.1. The aforementioned modifications to the standard AC-GAN architecture are shown on the left side of Figure 6.1.

The embedding network E is trained to learn a stochastic mapping with Gaussian noise rather than a deterministic mapping. This is because in the deterministic case, E can only use the stochasticity of the identity labels in the training set (which is fixed and typically very limited) to map the posterior distribution defined by $E(\boldsymbol{y})$ to the Gaussian prior distribution $p_{\boldsymbol{z}_{id}}$. Therefore, a deterministic mapping might not yield a smooth posterior distribution. In contrast, the additional randomness introduced by Gaussian noise in the stochastic mapping can alleviate this issue. With a stochastic mapping, the output of the embedding network E is a vector of means and variances that is used to produce samples that must be indistinguishable from samples from the Gaussian prior distribution $p_{\boldsymbol{z}_{id}}$. In order to allow backpropagation through the sampling operation, the reparametrization trick proposed in [48] is used.

To train the embedding network E and the discriminator network $D_{\boldsymbol{z}_{id}}$, the WGAN loss is used:

$$L_e = \mathbb{E}_{\boldsymbol{y}\sim p_y}[D_{\boldsymbol{z}_{id}}(E(\boldsymbol{y}))] - \mathbb{E}_{\boldsymbol{z}_{id}\sim p_{\boldsymbol{z}_{id}}}[D_{\boldsymbol{z}_{id}}(\boldsymbol{z}_{id})] + \lambda_e \, \mathbb{E}_{\tilde{\boldsymbol{y}}\sim p_{\tilde{\boldsymbol{y}}}}[\|(\nabla_{\tilde{\boldsymbol{y}}} D_{\boldsymbol{z}_{id}}(\tilde{\boldsymbol{y}})\|_2 - 1)^2] \tag{6.5}$$

where, in this case, $p_{\tilde{\boldsymbol{y}}}$ is a distribution of samples interpolated from the latent representation of the labels $E(\boldsymbol{y})$ and Gaussian samples $\boldsymbol{z}_{id}$. In a similar manner to Equation 6.3, λ_e is set to 10.

6.3.3 Mutual Information Loss

In the experiments, it was observed that the increased dimensionality of $\boldsymbol{z}_{id}$ with respect to the discrete labels $\boldsymbol{y}$ might cause some or all of the non-identity-related attributes to be encoded by $\boldsymbol{z}_{id}$ instead of $\boldsymbol{z}_{nid}$. For this reason, $\boldsymbol{z}_{nid}$ was forced to encode meaningful non-identity-related attributes by using a mutual information loss, as proposed in [249]. As mentioned in Section 6.2, this can be achieved through an auxiliary network D_{mi} in the discriminator D that is trained to predict $\boldsymbol{c} \subset \boldsymbol{z}_{nid}$. Since there is no prior knowledge about the latent variables $\boldsymbol{c}$, they are treated as

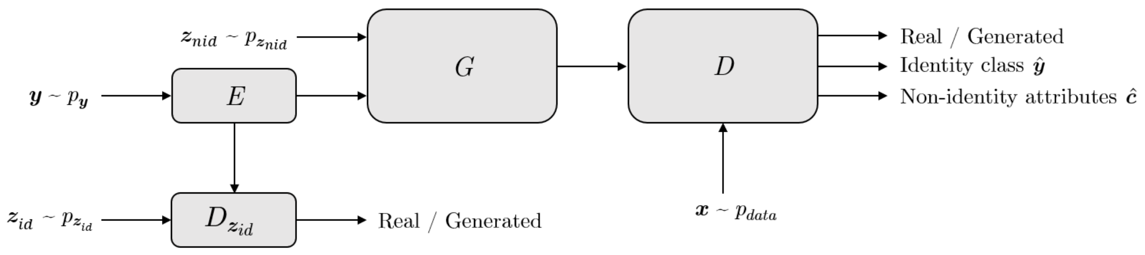


Figure 6.1: Proposed GAN model.

continuous variables and D_{mi} is trained as a regressor using minimum squared error (MSE):

$$L_{mi} = \mathbb{E}_{z_{nid} \sim p_{z_{nid}}, y \sim p_y}[\|c - D_{mi}(G(z_{nid} \mid E(y)))\|_2^2] \tag{6.6}$$

Balancing the mutual information loss from Equation 6.6 and the cross-entropy loss from Equation 6.4 is key to disentangling identity-related attributes from non-identity-related attributes.

6.3.4 Proposed GAN

Referring to Figure 6.1, it should be noted that, in practice, the auxiliary networks D_c and D_{mi} share all layers with the discriminator D. Hence, the last layer of D is split into three components that are trained with different loss functions (adversarial loss L_{img} for the real/generated classifier, cross-entropy loss L_c for the identity classifier and MSE loss L_{mi} for the non-identity-related attributes regressor). The architectures of the generator G and the discriminator D are the same as those in PGGAN [240]. The embedding network E contains an embedding layer that maps the discrete identity labels y to real-valued vectors, followed by two fully-connected layers. The discriminator network $D_{z_{id}}$ contains three fully-connected layers. Both the dimensionality of the latent variables z_{id} encoding the identity-related attributes and the dimensionality of the latent variables z_{nid} encoding the non-identity-related attributes are fixed to 64. In the experiments, no major difference was observed when making c a subset of z_{nid} or simply making c equal to z_{nid}. For simplicity, the latter option was adopted.

The proposed GAN is trained with the following overall loss:

$$L = L_{img} + \alpha L_c + \beta L_e + \gamma L_{mi} \tag{6.7}$$

where α, β and γ are weights controlling the contributions of L_c, L_e and L_{mi} to the loss relative to the contribution of L_{img}. Note that Equation 6.7 is the loss used when training the discriminators D and $D_{z_{id}}$. The generator G and the embedding network E are trained with the same loss as Equation 6.7 except that the adversarial losses L_{img} and L_e have a negative sign. After extensive experimentation,

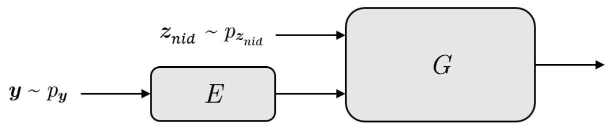


Figure 6.2: Generation of images of subjects $\boldsymbol{y}$ in the training set with identity-related attributes $E(\boldsymbol{y})$ and random non-identity-related attributes $\boldsymbol{z}_{nid}$.

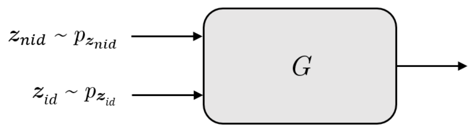


Figure 6.3: Generation of images of new subjects with random identity-related-attributes $\boldsymbol{z}_{id}$ and random non-identity-related attributes $\boldsymbol{z}_{nid}$.

the values $\alpha = 1$, $\beta = 1$ and $\gamma = 50$ were set. These weights are highly dependent on the specific architecture used in this work and should be tuned as necessary for different architectures.

Once the model is trained, multiple images of the same subject can be generated by feeding the generator with a fixed vector of identity-related attributes and different vectors of non-identity-related attributes. The model allows generation of images of subjects in the training set by feeding the generator with the latent representation of their labels $E(\boldsymbol{y})$ obtained by mapping $\boldsymbol{y}$ through E, as shown in Figure 6.2. Since $E(\boldsymbol{y})$ is trained to follow a Gaussian distribution $p_{\boldsymbol{z}_{id}}$, the generator can also be fed with a random sample $\boldsymbol{z}_{id}$ to generate images of new subjects, as shown in Figure 6.3.

6.4 Experiments

In this section, a qualitative analysis of the synthetic images generated by the proposed GAN is first provided. Next, the feasibility of augmenting face datasets with synthetic images, both in terms of width and depth, is explored. The augmented datasets are used to train CNN-based face recognition models (henceforth referred to as discriminative models) to determine whether they achieve a higher accuracy than models trained with real images alone.

The discriminative models used in this work consist of a popular CNN architecture based on residual blocks that has been used in other face recognition works [150, 153, 152]. The network is trained with softmax loss and optimised using stochastic gradient descent with momentum. The initial learning rate is set to 0.01 and decreased during training whenever the accuracy on the validation set stops improving. The input to the network are 100 $\times$ 100 RGB images. When training with datasets augmented with synthetic images it was ensured that on each training batch the number of real and synthetic images is roughly the same.

Three different subsets of the curated version of the VGGFace dataset [11] are used to train the discriminative models. The number of subjects and images of each subset is specified in Table 6.1. This dataset was selected because it contains a good number of images per subject, which helps the training of the proposed GAN. Moreover, the curated version does not contain mislabelled samples that might decrease the performance of the model.

6.4.1 Qualitative Analysis of Generated Images

The proposed GAN is first trained using the VGGFace$^{\text{large}}$ dataset. Figure 6.4 shows synthetic images of subjects in the training set generated by the trained model using the method shown in Figure 6.2. The identity-related attributes $E(\boldsymbol{y})$ have been fixed for each row and the non-identity-related attributes $\boldsymbol{z}_{nid}$ have been fixed for each column. The highlighted images in the first column are real images from the training set. Note how many of the synthetic images are as photo-realistic as the real images shown in the first column of Figure 6.4. Figure 6.5 shows synthetic images of new subjects generated by the trained model using the method shown in Figure 6.3. As in Figure 6.4, the identity-related attributes $\boldsymbol{z}_{id}$ have been fixed for each row and the non-identity-related attributes $\boldsymbol{z}_{nid}$ have been fixed for each column. Note how in both Figures 6.4 and 6.5, the images in each row appear to belong to the same subject since the identity-related attributes have been fixed. In contrast, the images in each column display common attributes that do not affect the identity of the subjects (e.g. head pose, facial expression and background) since the non-identity-related attributes have been fixed. From this it can be concluded that the proposed GAN is able to effectively disentangle identity-related attributes from non-identity-related attributes.

To test whether the proposed method can generate images of subjects not present in the training set, it was ensured that the synthetic images of new subjects indeed display identities that do not exist in the training set. Figures 6.6a to 6.6c show a comparison between synthetic images of three new subjects (shown in the top row of each of Figures 6.6a to 6.6c) and synthetic images of their most similar subject in the training set (shown in the bottom row of each of Figures 6.6a to 6.6c). These figures were created by measuring the average image difference between synthetic images of each new subject in Figures 6.6a to 6.6c and synthetic images of each subject in the training set. Since only the identity of the subjects needed to be compared, an average over image differences between synthetic images generated with the same non-identity-related attributes $\boldsymbol{z}_{nid}$ was done, as observed

Table 6.1: VGGFace subsets used for training the models.

Dataset	Number of subjects	Number of images
VGGFace$^{\text{large}}$	2558	734,665
VGGFace$^{\text{medium}}$	800	227,466
VGGFace$^{\text{small}}$	200	58,952

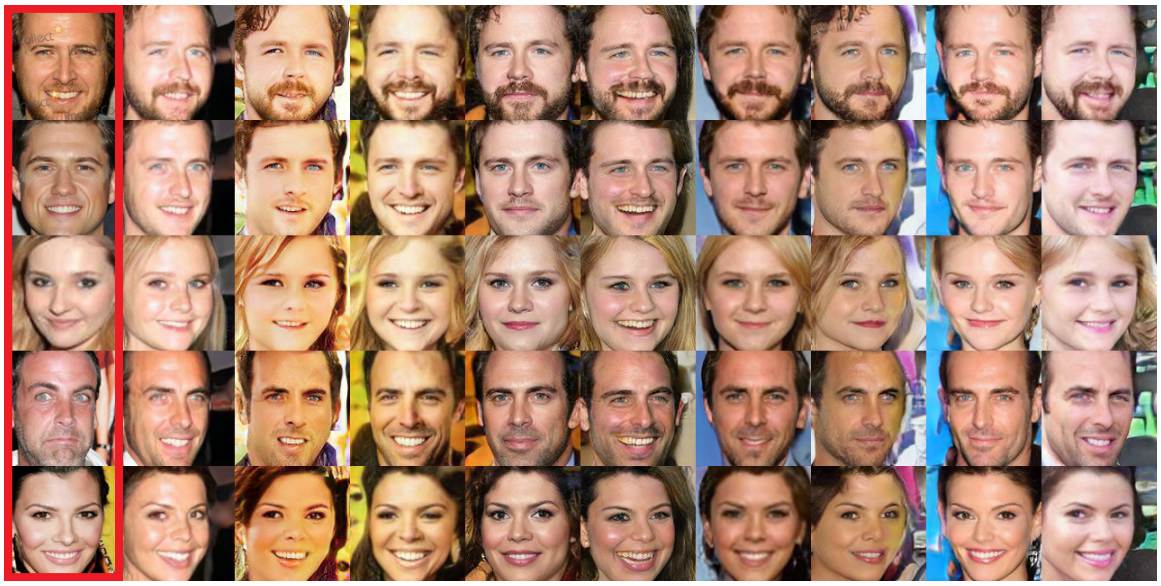


Figure 6.4: Synthetic images of subjects in the training set generated by the proposed GAN using the method shown in Figure 6.2. The identity related attributes have been fixed for each row and the non-identity related attributes have been fixed for each column. Note that the highlighted images in the first column are real images from the training set.

in the columns of each pair of rows in Figures 6.6a to 6.6c. It can be seen how even though the synthetic images of new subjects look similar to the synthetic images of their most similar subject in the training set, it is possible to visually differentiate them as two different identities. Hence, it can be concluded that the proposed GAN is able to successfully generate images of subjects not present in the training set.

It can also be shown how the model has not overfit the training images by applying linear interpolation between two random vectors of identity-related attributes $\boldsymbol{z}_{id}$ and two random vectors of non-identity-related attributes $\boldsymbol{z}_{nid}$. Figure 6.7 shows how the transition between synthetic images generated from interpolated vectors is smooth and visually consistent with what would be expected from mixing attributes of two face images. This suggests that the model is able to generate images with enough diversity and it is not just learning to replicate the training images.

6.4.2 Augmenting Datasets with Synthetic Images

In order to evaluate the quality of datasets augmented with synthetic images, several discriminative models were trained with different combinations of real and synthetic images and evaluated against models trained with real images alone. Each augmented dataset is created by adding synthetic images to one of the VGGFace subsets shown in Table 6.1. The synthetic images are generated

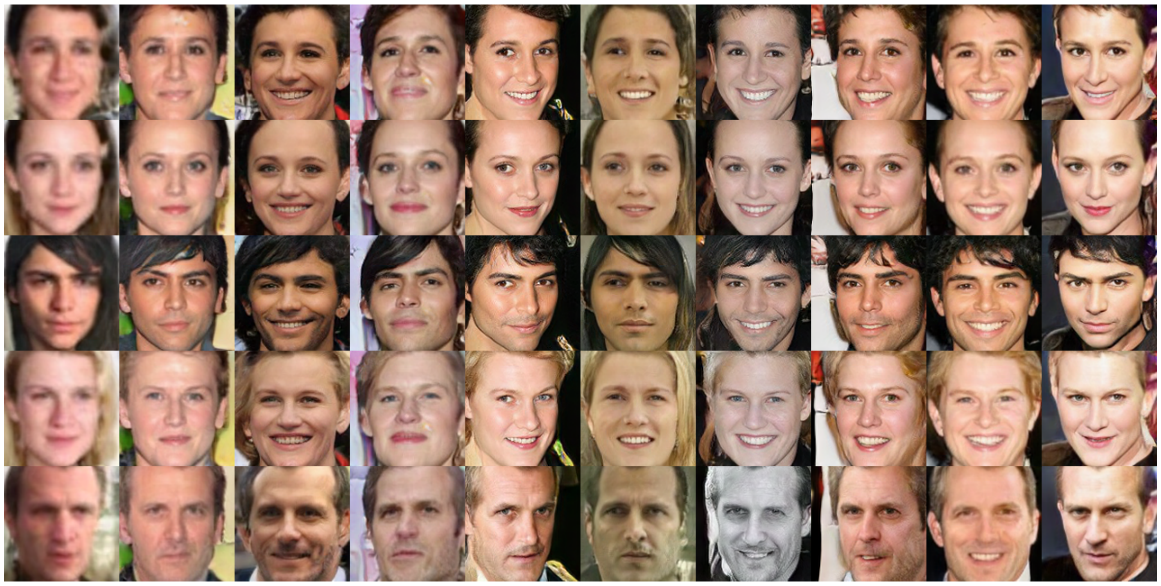


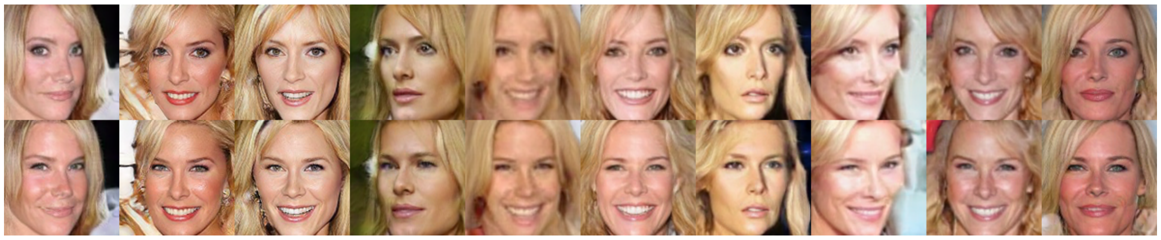
Figure 6.5: Synthetic images of new subjects generated by the proposed GAN using the method shown in Figure 6.3. The identity related attributes have been fixed for each row and the non-identity related attributes have been fixed for each column.

using the proposed GAN trained with the same dataset that was going to be augmented. For example, to augment VGGFace$^{\text{small}}$, synthetic images generated by the proposed GAN trained with VGGFace$^{\text{small}}$ are added to it. In this way, it can realistically be assessed whether the performance of a discriminative model can be improved by augmenting its training set using the proposed method. All the discriminative models are evaluated using the IJB-A dataset [252]. In particular, the verification protocol described in [252] is used and the true acceptance rate when the false acceptance rate is fixed to 0.01 is reported. The IJB-A dataset was chosen for evaluation because it contains challenging images that do not overlap with any of the subjects in the VGGFace dataset.

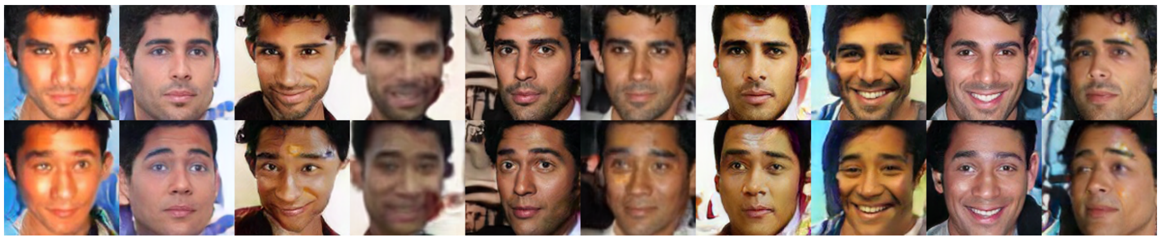
Table 6.2 shows the accuracy of models trained with depth-augmented datasets, i.e. datasets augmented by increasing the number of images per subject with synthetic images. For each subject

Table 6.2: Accuracy of discriminative models trained with depth-augmented datasets. The reported accuracy corresponds to the TAR@FAR=0.01 obtained when evaluating the models on the IJB-A dataset.

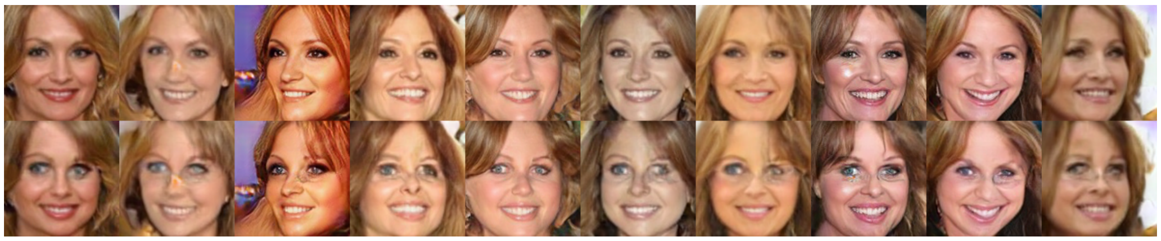
	Number of synthetic images per subject			
Training set	0	250	500	1000
VGGFace$^{\text{large}}$	67.58%	66.65%	**69.02%**	67.74%
VGGFace$^{\text{medium}}$	50.32%	52.25%	**52.54%**	51.97%
VGGFace$^{\text{small}}$	30.64%	33.30%	**35.15%**	32.95%



(a)



(b)



(c)

Figure 6.6: Comparison between synthetic images of new subjects and synthetic images of their most similar subject in the training set. The top row of each of (a), (b), (c) contains synthetic images of a new subject and the bottom row of each of (a), (b), (c) contains synthetic images of their most similar subject in the training set. Note that the non-identity-related attributes only vary across the rows of (a), (b), (c) to restrict the comparison to the identity of the subjects.

in the training set multiple synthetic images were generated by fixing the vector of identity-related attributes $E(\boldsymbol{y})$ and randomly sampling different vectors of non-identity-related attributes $\boldsymbol{z}_{nid}$. It can be seen how, in general, the accuracy of the models trained with depth-augmented datasets increases with respect to the models trained without synthetic images. In particular, maximum accuracy improvements of +1.44%, +2.22% and +4.51% were obtained when adding 500 synthetic images per subject to the VGGFace$^{\text{large}}$, VGGFace$^{\text{medium}}$ and VGGFace$^{\text{small}}$ datasets respectively. These results are consistent with the intuition that adding synthetic images to smaller datasets should result in greater improvement than adding them to larger datasets which contain more real images. The results shown in Table 6.2 also suggest that there is an optimal balance between the number of real and synthetic images per subject in a given dataset. Indeed, adding 1,000 synthetic images per subject to the VGGFace$^{\text{large}}$, VGGFace$^{\text{medium}}$ and VGGFace$^{\text{small}}$ datasets resulted in

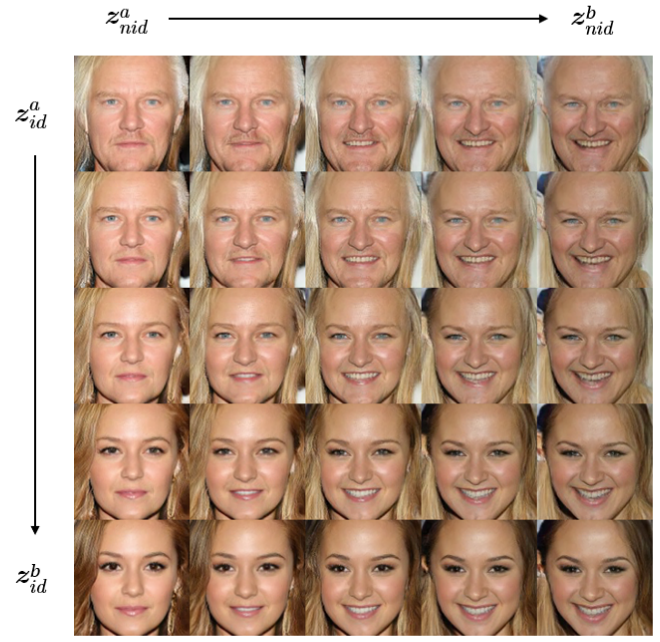


Figure 6.7: Synthetic images generated by interpolating between two random vectors of identity related attributes $\boldsymbol{z}^a_{id}$, $\boldsymbol{z}^b_{id}$ and two random vectors of non-identity related attributes $\boldsymbol{z}^a_{nid}$, $\boldsymbol{z}^b_{nid}$.

lower accuracy than adding 500 synthetic images per subject since the proportion of real images per subject becomes smaller. Table 6.3 shows the accuracy of models trained with width-augmented datasets, i.e. datasets augmented by increasing the number of subjects with synthetic images of new subjects. For each new subject, 500 synthetic images are generated (since this was the best number of synthetic images per subject obtained in Table 6.2) by fixing a randomly sampled vector of identity-related attributes $\boldsymbol{z}_{id}$ and randomly sampling different vectors of non-identity-related attributes $\boldsymbol{z}_{nid}$. Again, improvement is observed in most cases. In particular, a maximum accuracy improvement of +1.19% was obtained when adding 1,500 synthetic subjects to the VGGFace$^{\text{large}}$ dataset; and +4.23% and +8.42% when adding 1,000 synthetic subjects to the VGGFace$^{\text{medium}}$

Table 6.3: Accuracy of discriminative models trained with width-augmented datasets. The reported accuracy corresponds to the TAR@FAR=0.01 obtained when evaluating the models on the IJB-A dataset.

	Number of synthetic subjects			
Training set	0	500	1000	1500
VGGFace$^{\text{large}}$	67.58%	65.76%	66.32%	**68.77**%
VGGFace$^{\text{medium}}$	50.32%	52.15%	**54.55**%	54.32%
VGGFace$^{\text{small}}$	30.64%	38.06%	**39.06**%	38.81%

and VGGFace$^{\text{small}}$ datasets respectively. In this case, it can also be observed that adding synthetic images to the VGGFace$^{\text{large}}$ dataset does not significantly change the recognition accuracy. This can be explained by the fact that this dataset already contains a large number of real subjects. In contrast, a large improvement is observed when increasing the number of synthetic subjects in the VGGFace$^{\text{medium}}$ and VGGFace$^{\text{small}}$ datasets, as the number of real subjects in neither of these datasets is very large. It can also be observed that there seems to be an optimal balance between the number of real and synthetic subjects in a given dataset. Indeed, as shown in Table 6.3, adding 1,500 synthetic subjects to the VGGFace$^{\text{medium}}$ and VGGFace$^{\text{small}}$ does not result in higher accuracy than adding 1,000 synthetic subjects since the proportion of real subjects becomes smaller. Note that in the case of the VGGFace$^{\text{large}}$ dataset, more synthetic subjects can be added since there is still a good balance between real and synthetic subjects. However, as mentioned earlier, this dataset already contains a large number of real subjects. Hence, it is expected that no significant improvement in recognition accuracy will be obtained by adding more synthetic subjects.

Looking at the results shown in Tables 6.2 and 6.3, it can be concluded that augmenting datasets with synthetic images is mainly beneficial for small and medium datasets. Moreover, the accuracy improvement obtained when training with width-augmented and depth-augmented datasets is relative to the number of subjects and number of images per subject of each augmented dataset. For example, the VGGFace$^{\text{small}}$ dataset contains 200 subjects and an average of 295 images per subject. Thus, it is reasonable that the accuracy is improved by a larger margin when adding synthetic images of new subjects (+8.42%) than when adding synthetic images of existing subjects (+4.51%).

6.5 Conclusions

This chapter has studied the feasibility of augmenting face datasets with photo-realistic synthetic images. In particular, a new type of conditional GAN that can generate photo-realistic face images from two latent vectors encoding identity-related attributes and non-identity-related attributes respectively has been presented. By fixing the latent vector of identity-related attributes and varying the latent vector of non-identity-related attributes, the proposed GAN can generate images of subjects with fixed identities but different attributes, such as facial expression and head pose. The introduction of an embedding network to map discrete identity labels to a continuous latent space of identities enables the generation of both images of subjects in the training set and images of new subjects not in the training set. The experiments have shown the effectiveness of the disentangled representation and the high visual quality of the generated images. To demonstrate the benefit of augmenting datasets with the proposed method, several CNN-based face recognition models with different combinations of real and synthetic images were trained. In most cases, the discriminative models trained with a combination of real and synthetic images have outperformed the discriminative models trained with real images alone. According to the experimental results, the proposed method is particularly effective when augmenting datasets with a moderate number of subjects and/or images per subject.

Compared to other face image synthesis methods explicitly designed to generate face images, the proposed method is more generic and can be used to generate any kind of images. Moreover, by only adding a few simple modifications to the standard AC-GAN architecture, the proposed method can be easily extended. For example, the proposed GAN could be extended to control the non-identity-related attributes explicitly by conditioning the GAN on specific attributes. This would allow face datasets to be augmented in a tailored manner, e.g. by adding synthetic images of subjects with sunglasses, facial hair, different ages, etc. Hopefully, this and other ideas derived from this work will contribute to the development of new data augmentation techniques.

Chapter 7

Conclusions

This thesis has presented several methods to improve the accuracy of face recognition algorithms in real-world applications.

First, Chapter 4 introduced a novel traditional face recognition method based on a combination of hand-engineered features and subspace representation techniques. This method was applied to the task of comparing face images printed on ID documents to face images stored in the biometric chip of such documents. By training with a very small dataset with only 1,000 images of 500 subjects, the proposed method achieved very high accuracy on this task. A second evaluation performed on a more generic face dataset confirmed the efficacy of the proposed method when dealing with frontal face images. This work showed that traditional face recognition methods can still be relevant in situations in which training data and/or computational resources are limited. In Chapter 4, it was also shown the benefits of using two operating points instead of one when providing the algorithm's final verification decision. This simple modification can be incorporated into any face verification system to better control the error rates at the expense of having cases in which the result is undetermined (which can be handled by human operators, for example).

Next, the recognition of faces in-the-wild using convolutional neural networks was considered. In particular, Chapter 5 proposed a method to improve the performance of CNN-based face recognition methods when dealing with facial occlusions. The proposed approach consisted of artificially occluding the face regions that are most sensitive to occlusions during training. This method was shown to not only improve the performance when dealing with faces with occlusions but also with faces without occlusions. Chapter 5 also proposed a new metric learning loss function that improves the performance of the commonly used triplet loss function. Fine-tuning models with this loss function was shown to further improve face recognition accuracy. This work showed that the performance of CNN-based face recognition models can be improved without collecting more training data.

Finally, Chapter 6 investigated the feasibility of using synthetic data to improve the performance of CNN-based face recognition models. More specifically, a generative model that allows the generation of both face images of existing subjects in a training set and face images of new subjects was proposed. By using the latest techniques in generative adversarial networks, the proposed method was able to generate photo-realistic images often indistinguishable from real images. It was demonstrated how the accuracy of CNN-based face recognition models can be improved by training with datasets augmented with synthetic face images generated by the proposed method. This work showed that the generation of synthetic training data is a promising alternative to the expensive and laborious task of collecting real training data.

Face recognition has followed the same transition as many other computer vision applications. Traditional methods based on hand-engineered features that provided state-of-the-art accuracy only a few years ago have been replaced by methods based on deep learning, provided that enough training data and computational resources are available. This has raised the issue of collecting good quality, large-scale datasets, which in many cases is a very challenging task. It is clear then that new techniques must be developed to reduce the need for such datasets. Achieving this will require the development of new neural networks architectures, loss functions and data augmentation techniques. It is worth noting that these trends are not unique to face recognition. In particular, the search for better neural network architectures has given rise to a family of methods known as *AutoML*[1] that use machine learning techniques to find more efficient neural network architectures; and unsupervised and semi-supervised learning methods are a very active area of research that aim to provide an alternative to the expensive supervised learning paradigm that currently dominates the real-world applicability of deep learning. Both of these will directly contribute to the development of face recognition models that require less data and computational resources to be trained and deployed.

Looking back at the work presented in this thesis, some future work can be proposed. The generative model proposed in Chapter 6 could be extended to generate face images with controllable attributes such as age, occlusion, head pose, illumination, etc. This could be achieved by conditioning the generative model with extra labels representing the presence of each of those attributes. Having such a generative model would allow the extension of the idea presented in Chapter 5. In particular, an analysis similar to the occlusion sensitivity experiment done in Chapter 5 could be done for each of the face attributes separately, and the generative model could then be used to generate photo-realistic face images with those attributes as required. For example, a sensitivity analysis for age variation could be done by simply testing the model with face images of subjects of different ages. Then, during training, the generative model could be used to generate face images of subjects of the most sensitive ages more frequently than the least sensitive ages. This idea could be further extended to work dynamically so that a sensitivity analysis for each attribute is periodically performed during training using the partially trained model. In this way, face images with the most sensitive attributes could be generated and added to the training set until the next time the sensitivity analysis for each attribute is performed. This kind of smart data augmentation scheme would allow training face

[1]A list of papers related to AutoML approaches can be found online at `https://www.automl.org/automl/literature-on-neural-architecture-search/`

recognition models with datasets that dynamically become more balanced and diverse when needed. Another possible improvement related to this thesis would be to combine the batch triplet loss presented in Chapter 5 with one of the softmax losses with margin discussed at the end of Chapter 3. This could be achieved by either training with both losses (at the same time or alternating them during training) or by first training using softmax loss with margin and then fine-tuning with batch triplet loss, similar to the approach followed in Chapter 5.

Overall, the progress made in face recognition research in the last few years has been unprecedented, as evidenced by the widely adoption of face recognition technology by many business and organisations. State-of-the-art algorithms are now used in many real-world applications and are no longer seen as purely academic achievements. As shown in this thesis, many challenges arise when dealing with the in-the-wild face images that are typically found in real-world applications. It is expected that tackling these challenges will allow face recognition algorithms to surpass human-level accuracy in the near future. Hopefully, the methods developed in this thesis will contribute towards that goal.

Bibliography

[1] M. D. Kelly, *Visual identification of people by computer.* PhD thesis, Stanford University, 1970.

[2] T. Kanade, *Picture processing by computer complex and recognition of human faces.* PhD thesis, Kyoto University, 1973.

[3] K. Delac and M. Grgic, "A survey of biometric recognition methods," in *Proceedings of the 46th International Symposium on Electronics in Marine*, vol. 46, pp. 16–18, 2004.

[4] U. Park, Y. Tong, and A. K. Jain, "Age-invariant face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 947–954, 2010.

[5] Z. Li, U. Park, and A. K. Jain, "A discriminative model for age invariant face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1028–1037, 2011.

[6] C. Ding and D. Tao, "A comprehensive survey on pose-invariant face recognition," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 3, p. 37, 2016.

[7] D.-H. Liu, K.-M. Lam, and L.-S. Shen, "Illumination invariant face recognition," *Pattern Recognition*, vol. 38, no. 10, pp. 1705–1716, 2005.

[8] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1635–1650, 2010.

[9] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1891–1898, 2014.

[10] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[11] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, "Deep face recognition.," in *Proceedings of the British Machine Vision Conference*, pp. 41.1–41.12, 2015.

[12] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "MS-Celeb-1M: A dataset and benchmark for large-scale face recognition," in *Proceedings of the European Conference on Computer Vision*, pp. 87–102, 2016.

[13] A. Nech and I. Kemelmacher-Shlizerman, "Level playing field for million scale face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3406–3415, 2017.

[14] A. Bansal, A. Nanduri, C. D. Castillo, R. Ranjan, and R. Chellappa, "UMDFaces: An annotated face dataset for training deep networks," in *Proceedings of the IEEE International Joint Conference on Biometrics*, pp. 464–473, 2017.

[15] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 67–74, 2018.

[16] D. Sáez-Trigueros, H. Hertlein, L. Meng, and M. Hartnett, "Shape and texture combined face recognition for detection of forged id documents," in *Proceedings of the 39th IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics*, pp. 1343–1348, 2016.

[17] D. S. Trigueros, L. Meng, and M. Hartnett, "Enhancing convolutional neural networks for face recognition with occlusion maps and batch triplet loss," *Image and Vision Computing*, vol. 79, pp. 99–108, 2018.

[18] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," *presented at a talk at the Stanford Artificial Project*, 1968.

[19] L. G. Roberts, *Machine perception of three-dimensional solids.* PhD thesis, Massachusetts Institute of Technology, 1963.

[20] G. J. Agin and T. O. Binford, "Computer description of curved objects," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pp. 629–640, 1973.

[21] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 67–92, 1973.

[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[23] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[24] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[25] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Local binary features for texture classification: taxonomy and experimental study," *Pattern Recognition*, vol. 62, pp. 135–160, 2017.

[26] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[27] A. Gunay and V. V. Nabiyev, "Automatic age classification with LBP," in *Proceedings of the 23rd International Symposium on Computer and Information Sciences*, pp. 1–4, 2008.

[28] X. Wang, H. Gong, H. Zhang, B. Li, and Z. Zhuang, "Palmprint identification using boosting local binary pattern," in *Proceedings of the International Conference On Pattern Recognition, 2006*, vol. 3, pp. 503–506, 2006.

[29] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[30] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision*, pp. 404–417, 2006.

[31] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2548–2555, 2011.

[32] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, pp. 312–317, 1996.

[33] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[34] T. Bouwmans, C. Silva, C. Marghes, M. S. Zitouni, H. Bhaskar, and C. Frelicot, "On the role and the importance of features for background modeling and foreground detection," *Computer Science Review*, vol. 28, pp. 26–91, 2018.

[35] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[36] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[37] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[38] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[39] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[41] A. Krogh and J. A. Hertz, “A simple weight decay can improve generalization,” in *Advances in Neural Information Processing Systems*, pp. 950–957, 1991.

[42] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[44] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[45] Y. LeCun *et al.*, “Generalization and network design strategies,” in *Connectionism in Perspective*, vol. 19, Elsevier, 1989.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

[48] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proceedings of the International Conference on Learning Representations*, 2014.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[50] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1747–1756, 2016.

[51] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio,” in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

[52] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the International Conference on Machine Learning*, pp. 214–223, 2017.

[53] T. Hassner, S. Harel, E. Paz, and R. Enbar, "Effective face frontalization in unconstrained images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4295–4304, 2015.

[54] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[55] P. Grother, R. J. Micheals, and P. J. Phillips, "Face recognition vendor test 2002 performance metrics," in *Proceedings of the International Conference on Audio- and Video-based Biometric Person Authentication*, pp. 937–945, 2003.

[56] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[57] J. Shi, A. Samal, and D. Marx, "How effective are landmarks and their geometry for face recognition?," *Computer Vision and Image Understanding*, vol. 102, no. 2, pp. 117–133, 2006.

[58] I. L. Dryden and K. V. Mardia, *Statistical shape analysis*, vol. 4. Wiley Chichester, 1998.

[59] F. Daniyal, P. Nair, and A. Cavallaro, "Compact signatures for 3D face recognition under varying expressions," in *Proceedings of 6th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 302–307, 2009.

[60] S. Gupta, M. K. Markey, and A. C. Bovik, "Anthropometric 3D face recognition," *International Journal of Computer Vision*, vol. 90, no. 3, pp. 331–349, 2010.

[61] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America A*, vol. 4, no. 3, pp. 519–524, 1987.

[62] M. Kirby and L. Sirovich, "Application of the karhunen-loeve procedure for the characterization of human faces," *IEEE Transactions on Pattern analysis and Machine intelligence*, vol. 12, no. 1, pp. 103–108, 1990.

[63] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[64] B. Moghaddam, W. Wahid, and A. Pentland, "Beyond eigenfaces: Probabilistic matching for face recognition," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 30–35, 1998.

[65] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International Conference on Artificial Neural Networks*, pp. 583–588, 1997.

[66] K. I. Kim, K. Jung, and H. J. Kim, "Face recognition using kernel principal component analysis," *IEEE Signal Processing Letters*, vol. 9, no. 2, pp. 40–42, 2002.

[67] P. Comon, "Independent component analysis, a new concept?," *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[68] M. S. Bartlett, "Independent component representations for face recognition," in *Face Image Analysis by Unsupervised Learning*, pp. 39–67, Springer, 2001.

[69] J. Yang, D. Zhang, A. F. Frangi, and J.-y. Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[70] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, pp. 138–142, 1994.

[71] R. A. Fisher, "The statistical utilization of multiple measurements," *Annals of Human Genetics*, vol. 8, no. 4, pp. 376–386, 1938.

[72] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[73] K. Etemad and R. Chellappa, "Discriminant analysis for recognition of human face images," *Journal of the Optical Society of America A*, vol. 14, no. 8, pp. 1724–1733, 1997.

[74] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng, "Discriminant analysis of principal components for face recognition," in *Face Recognition*, pp. 73–85, Springer, 1998.

[75] W. Zhao, R. Chellappa, and P. J. Phillips, "Subspace linear discriminant analysis for face recognition," tech. rep., Center for Automation Research CAR-TR-914, University of Maryland, 1999.

[76] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing IX: Proceedings of the IEEE Signal Processing Society Workshop*, pp. 41–48, 1999.

[77] Q. Liu, R. Huang, H. Lu, and S. Ma, "Face recognition using kernel-based fisher discriminant analysis," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 197–201, 2002.

[78] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proceedings of the European Conference on Computer Vision*, pp. 531–542, 2006.

[79] P. J. Phillips, “Support vector machines applied to face recognition,” in *Advances in Neural Information Processing Systems*, pp. 803–809, 1999.

[80] K. Jonsson, J. Kittler, Y. Li, and J. Matas, “Support vector machines for face authentication,” *Image and Vision Computing*, vol. 20, no. 5-6, pp. 369–375, 2002.

[81] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, “Face recognition using laplacianfaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328–340, 2005.

[82] D. Cai, X. He, J. Han, and H.-J. Zhang, “Orthogonal laplacianfaces for face recognition,” *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3608–3614, 2006.

[83] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[84] Q. Zhang and B. Li, “Discriminative K-SVD for dictionary learning in face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2691–2698, 2010.

[85] Z. Zhou, A. Wagner, H. Mobahi, J. Wright, and Y. Ma, “Face recognition with contiguous occlusion using markov random fields,” in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 1050–1057, 2009.

[86] H. Jia and A. M. Martinez, “Face recognition with occlusions in the training and testing sets,” in *Proceedings of the 8th IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 1–6, 2008.

[87] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, “Bayesian face revisited: A joint formulation,” in *Proceedings of the European Conference on Computer Vision*, pp. 566–579, 2012.

[88] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” tech. rep., pp. 07-49, University of Massachusetts, Amherst, 2007.

[89] A. Pentland, B. Moghaddam, T. Starner, *et al.*, “View-based and modular eigenspaces for face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[90] B. Takacs, “Comparing face images using the modified hausdorff distance,” *Pattern Recognition*, vol. 31, no. 12, pp. 1873–1881, 1998.

[91] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[92] Y. Gao and M. K. Leung, "Face recognition using line edge map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 764–779, 2002.

[93] L. Wiskott, N. Krüger, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[94] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. Von Der Malsburg, R. P. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 300–311, 1993.

[95] T. S. Lee, "Image representation using 2D gabor wavelets," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 959–971, 1996.

[96] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, vol. 12, pp. 296–301, 1995.

[97] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[98] A. Albiol, D. Monzo, A. Martin, J. Sastre, and A. Albiol, "Face recognition using HOG–EBGM," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1537–1543, 2008.

[99] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[100] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, "Local binary patterns and its application to facial image analysis: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 6, pp. 765–781, 2011.

[101] W. Zhang, S. Shan, H. Zhang, W. Gao, and X. Chen, "Multi-resolution histograms of local variation patterns (MHLVP) for robust face recognition," in *International Conference on Audio- and Video-Based Biometric Person Authentication*, pp. 937–944, 2005.

[102] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition," in *Proceedings of the 10th International Conference on Computer Vision*, vol. 1, pp. 786–791, 2005.

[103] T. Ahonen, J. Matas, C. He, and M. Pietikäinen, "Rotation invariant image description with local binary pattern histogram fourier features," in *Proceedings of the Scandinavian Conference on Image Analysis*, pp. 61–70, 2009.

[104] B. Zhang, Y. Gao, S. Zhao, and J. Liu, "Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 533–544, 2010.

[105] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli, "On the use of SIFT features for face authentication," in *Proceedings of the Computer Vision and Pattern Recognition Workshop*, pp. 35–35, 2006.

[106] P. Dreuw, P. Steingrube, H. Hanselmann, H. Ney, and G. Aachen, "SURF-Face: Face recognition under viewpoint consistency constraints.," in *Proceedings of the British Machine Vision Conference*, pp. 1–11, 2009.

[107] C. Geng and X. Jiang, "Face recognition using SIFT features," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 3313–3316, 2009.

[108] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face recognition with learning-based descriptor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2707–2714, 2010.

[109] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[110] Y. Freund, S. Dasgupta, M. Kabra, and N. Verma, "Learning the structure of manifolds using random projections," in *Advances in Neural Information Processing Systems*, pp. 473–480, 2008.

[111] G. Sharma, S. ul Hussain, and F. Jurie, "Local higher-order statistics (LHS) for texture categorization and facial analysis," in *Proceedings of the European Conference on Computer Vision*, pp. 1–12, 2012.

[112] Z. Lei, M. Pietikäinen, and S. Z. Li, "Learning discriminant face descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 289–302, 2014.

[113] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image processing*, vol. 11, no. 4, pp. 467–476, 2002.

[114] C. Liu and H. Wechsler, "Independent component analysis of gabor features for face recognition," *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 919–928, 2003.

[115] C. Liu, "Gabor-based kernel PCA with fractional power polynomial models for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 572–581, 2004.

[116] C. Liu and H. Wechsler, "Robust coding schemes for indexing and retrieval from large face databases," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 132–137, 2000.

[117] C.-H. Chan, J. Kittler, and K. Messer, "Multi-scale local binary pattern histograms for face recognition," in *International Conference on Biometrics*, pp. 809–818, 2007.

[118] C.-H. Chan, J. Kittler, and K. Messer, "Multispectral local binary pattern histogram for component-based color face verification," in *Proceedings of the 1st IEEE International Conference on Biometrics: Theory, Applications, and Systems*, pp. 1–7, 2007.

[119] D. Zhao, Z. Lin, and X. Tang, "Laplacian PCA and its applications," in *Proceedings of the 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[120] L. Wolf, T. Hassner, and Y. Taigman, "Descriptor based methods in the wild," in *Workshop on Faces in Real-Life Images at European Conference on Computer Vision*, 2008.

[121] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3025–3032, 2013.

[122] C. Lu and X. Tang, "Surpassing human-level face verification performance on LFW with gaussianface," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 3811–3819, 2015.

[123] R. Urtasun and T. Darrell, "Discriminative gaussian process latent variable model for classification," in *Proceedings of the 24th International Conference on Machine Learning*, pp. 927–934, 2007.

[124] X. Tan and B. Triggs, "Fusing gabor and LBP feature sets for kernel-based face recognition," in *International Workshop on Analysis and Modeling of Faces and Gestures*, pp. 235–249, 2007.

[125] H. Cevikalp, M. Neamtu, and M. Wilkes, "Discriminative common vector method with kernels," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1550–1565, 2006.

[126] S. Shan, W. Zhang, Y. Su, X. Chen, and W. Gao, "Ensemble of piecewise FDA based on spatial histograms of local (gabor) binary patterns for face recognition," in *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 3, 2006.

[127] C. H. Chan, M. A. Tahir, J. Kittler, and M. Pietikäinen, "Multiscale local phase quantization for robust component-based face recognition using kernel fusion of multiple descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1164–1177, 2013.

[128] E. Rahtu, J. Heikkilä, V. Ojansivu, and T. Ahonen, "Local phase quantization for blur-insensitive image analysis," *Image and Vision Computing*, vol. 30, no. 8, pp. 501–512, 2012.

[129] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[130] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[131] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 365–372, 2009.

[132] T. Berg and P. N. Belhumeur, "Tom-vs-pete classifiers and identity-preserving alignment for face verification.," in *Proceedings of the British Machine Vision Conference*, vol. 2, p. 7, 2012.

[133] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? metric learning approaches for face identification," in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 498–505, 2009.

[134] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[135] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 539–546, 2005.

[136] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou, "Learning deep face representation," *arXiv preprint arXiv:1403.2802*, 2014.

[137] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

[138] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, "Face recognition/detection by probabilistic decision-based neural network," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 114–132, 1997.

[139] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[140] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[141] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.

[142] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[143] K. Gregor and Y. LeCun, "Emergence of complex-like cells in a temporal product network with local receptive fields," *arXiv preprint arXiv:1006.0448*, 2010.

[144] G. B. Huang, H. Lee, and E. Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2518–2525, 2012.

[145] E. Zhou, Z. Cao, and Q. Yin, "Naive-deep face recognition: Touching the limit of LFW benchmark or not?," *arXiv preprint arXiv:1501.04690*, 2015.

[146] A. Bansal, C. D. Castillo, R. Ranjan, and R. Chellappa, "The do's and don'ts for CNN-based face verification.," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2545–2554, 2017.

[147] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.

[148] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[149] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[150] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *arXiv preprint arXiv:1703.09507*, 2017.

[151] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2017.

[152] Y. Wu, H. Liu, J. Li, and Y. Fu, "Deep face recognition with center invariant loss," in *Proceedings of the on Thematic Workshops of ACM Multimedia*, pp. 408–414, 2017.

[153] M. A. Hasnat, J. Bohné, J. Milgram, S. Gentric, and L. Chen, "DeepVisage: Making face recognition simple yet with powerful generalization skills.," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1682–1691, 2017.

[154] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "CosFace: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

[155] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[156] J. Deng, J. Guo, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," *arXiv preprint arXiv:1801.07698*, 2018.

[157] Y. Yamada, M. Iwamura, and K. Kise, "Deep pyramidal residual networks with separated stochastic depth," *arXiv preprint arXiv:1612.01230*, 2016.

[158] X. Wu, R. He, and Z. Sun, "A lightened CNN for deep face representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 4, 2015.

[159] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in Neural Information Processing Systems*, pp. 1988–1996, 2014.

[160] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2892–2900, 2015.

[161] Y. Sun, D. Liang, X. Wang, and X. Tang, "DeepID3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[162] J.-C. Chen, V. M. Patel, and R. Chellappa, "Unconstrained face verification using deep CNN features," in *Proceedings of IEEE Winter Conference on the Applications of Computer Vision*, pp. 1–9, 2016.

[163] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[164] S. Sankaranarayanan, A. Alavi, and R. Chellappa, "Triplet similarity embedding for face verification," *arXiv preprint arXiv:1602.03418*, 2016.

[165] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa, "Triplet probabilistic embedding for face verification and clustering," in *Proceedings of the IEEE 8th International Conference on Biometrics: Theory, Applications and Systems*, pp. 1–8, 2016.

[166] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proceedings of the European Conference on Computer Vision*, pp. 499–515, 2016.

[167] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long-tailed training data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5409–5418, 2017.

[168] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in *Proceedings of the 33rd International Conference on Machine Learning*, pp. 507–516, 2016.

[169] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of the International Conference on Machine Learning*, pp. 1558–1566, 2016.

[170] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional GANs for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[171] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," *Proceedings of the International Conference on Learning Representations*, 2016.

[172] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1225–1233, 2017.

[173] Y. Lu, Y.-W. Tai, and C.-K. Tang, "Conditional CycleGAN for attribute guided face image generation," *arXiv preprint arXiv:1705.09966*, 2017.

[174] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 2018.

[175] W. Yin, Y. Fu, L. Sigal, and X. Xue, "Semi-latent GAN: Learning to generate and modify facial images from attributes," *arXiv preprint arXiv:1704.02166*, 2017.

[176] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras, "Neural face editing with intrinsic image disentangling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5444–5453, 2017.

[177] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, *et al.*, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems*, pp. 5969–5978, 2017.

[178] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Arbitrary facial attribute editing: Only change what you want," *arXiv preprint arXiv:1711.10678*, 2017.

[179] Y. Zhou and B. E. Shi, "Photorealistic facial expression synthesis by the conditional difference adversarial autoencoder," in *Proceedings of the 7th International Conference on Affective Computing and Intelligent Interaction*, pp. 370–376, 2017.

[180] H. Ding, K. Sricharan, and R. Chellappa, "ExprGAN: Facial expression editing with controllable expression intensity," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[181] C. Donahue, A. Balsubramani, J. McAuley, and Z. C. Lipton, "Semantically decomposing the latent spaces of generative adversarial networks," in *Proceedings of the International Conference on Learning Representations*, 2018.

[182] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2439–2448, 2017.

[183] L. Q. Tran, X. Yin, and X. Liu, "Representation learning by rotating your faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[184] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 2089–2093, 2017.

[185] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017.

[186] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[187] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[188] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices," *Proceedings of the Chinese Conference on Biometric Recognition*, 2018.

[189] "Guidance on examining identity documents." UK Home Office, 2016.

[190] "Pre-employment screening - document verification." Centre for the Protection of National Infrastructure, 2015.

[191] T. Bourlai, A. Ross, and A. Jain, "On matching digital face images against scanned passport photos," in *Proceedings of the 1st IEEE International Conference on Biometrics, Identity and Security*, 2009.

[192] T. Bourlai, A. Ross, and A. K. Jain, "Restoring degraded face images: A case study in matching faxed, printed, and scanned photos," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 371–384, 2011.

[193] P. J. Phillips, J. R. Beveridge, B. A. Draper, G. Givens, A. J. O'Toole, D. S. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, and S. Weimer, "An introduction to the good, the bad, & the ugly face recognition challenge problem," in *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 346–353, 2011.

[194] T. Baltrusaitis, P. Robinson, and L.-P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 354–361, 2013.

[195] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models," in *Proceedings of the British Machine Vision Conference*, vol. 1, p. 3, 2006.

[196] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[197] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[198] J. C. Klontz, B. F. Klare, S. Klum, A. K. Jain, and M. J. Burge, "Open source biometric recognition," in *Proceedings of the 6th International Conference on Biometrics: Theory, Applications and Systems*, pp. 1–8, 2013.

[199] B. Klare and A. K. Jain, "Face recognition across time lapse: On learning feature subspaces," in *Proceedings of International Joint Conference on Biometrics*, pp. 1–8, 2011.

[200] B. F. Klare, M. J. Burge, J. C. Klontz, R. W. V. Bruegge, and A. K. Jain, "Face recognition performance: Role of demographic information," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1789–1801, 2012.

[201] X. Wang and X. Tang, "Random sampling for subspace face recognition," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 91–104, 2006.

[202] C. Liu and H. Wechsler, "A shape-and texture-based enhanced fisher classifier for face recognition," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 598–608, 2001.

[203] S. A. Rizvi, P. J. Phillips, and H. Moon, "The FERET verification testing protocol for face recognition algorithms," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 48–53, 1998.

[204] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 947–954, 2005.

[205] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision*, pp. 818–833, 2014.

[206] A. M. Martinez and R. Benavente, "The AR face database," tech. rep. CVC Technical Report 24, 1998.

[207] R. Min, A. Hadid, and J.-L. Dugelay, "Improving the recognition of faces occluded by facial accessories," in *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 442–447, 2011.

[208] M. Sharma, S. Prakash, and P. Gupta, "An efficient partial occluded face recognition system," *Neurocomputing*, vol. 116, pp. 231–241, 2013.

[209] S. Park, H. Lee, J.-H. Yoo, G. Kim, and S. Kim, "Partially occluded facial image retrieval based on a similarity measurement," *Mathematical Problems in Engineering*, 2015.

[210] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[211] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 341–349, 2012.

[212] L. Cheng, J. Wang, Y. Gong, and Q. Hou, "Robust deep auto-encoder for occluded face recognition," in *Proceedings of the 23rd ACM International Conference on Multimedia*, pp. 1099–1102, 2015.

[213] Y. Zhang, R. Liu, S. Zhang, and M. Zhu, "Occlusion-robust face recognition using iterative stacked denoising autoencoder," in *International Conference on Neural Information Processing*, pp. 352–359, 2013.

[214] F. Zhao, J. Feng, J. Zhao, W. Yang, and S. Yan, "Robust LSTM-autoencoders for face de-occlusion in the wild," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 778–790, 2018.

[215] J. Liu, Y. Deng, T. Bai, Z. Wei, and C. Huang, "Targeting ultimate accuracy: Face recognition via deep embedding," *arXiv preprint arXiv:1506.07310*, 2015.

[216] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393, 2014.

[217] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, 2015.

[218] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *Proceedings of the European Conference on Computer Vision*, pp. 241–257, 2016.

[219] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3109–3118, 2015.

[220] B. Kumar, G. Carneiro, I. Reid, *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5385–5394, 2016.

[221] J. Daugman, "Biometric decision landscapes," tech. rep., University of Cambridge, Computer Laboratory, 2000.

[222] G. B. Huang and E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," tech. rep., pp. 14–003, University of Massachusetts, Amherst, 2014.

[223] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2Image: Conditional image generation from visual attributes," in *Proceedings of the European Conference on Computer Vision*, pp. 776–791, 2016.

[224] B. Meden, R. C. Mallı, S. Fabijan, H. K. Ekenel, V. Štruc, and P. Peer, "Face deidentification with generative deep neural networks," *IET Signal Processing*, vol. 11, no. 9, pp. 1046–1054, 2017.

[225] B. Meden, Ž. Emeršič, V. Štruc, and P. Peer, "k-same-net: k-anonymity with generative deep neural networks for face deidentification," *Entropy*, vol. 20, no. 1, p. 60, 2018.

[226] K. Brkic, I. Sikiric, T. Hrkac, and Z. Kalafatic, "I know that person: Generative full body and face de-identification of people in images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1319–1328, 2017.

[227] Y. Wu, F. Yang, and H. Ling, "Privacy-protective-GAN for face de-identification," *arXiv preprint arXiv:1806.08906*, 2018.

[228] I. Masi, A. T. Tran, T. Hassner, J. T. Leksut, and G. Medioni, "Do we really need to collect millions of faces for effective face recognition?," in *Proceedings of the European Conference on Computer Vision*, pp. 579–596, 2016.

[229] S. Banerjee, J. S. Bernhard, W. J. Scheirer, K. W. Bowyer, and P. J. Flynn, "SREFI: Synthesis of realistic example face images," in *Proceedings of the IEEE International Joint Conference on Biometrics*, pp. 37–45, 2017.

[230] M. Osadchy, Y. Wang, O. Dunkelman, S. Gibson, J. Hernandez-Castro, and C. Solomon, "GenFace: Improving cyber security using realistic synthetic face generation," in *Proceedings of the International Conference on Cyber Security Cryptography and Machine Learning*, pp. 19–33, 2017.

[231] I. Masi, F.-J. Chang, J. Choi, S. Harel, J. Kim, K. Kim, J. Leksut, S. Rawls, Y. Wu, T. Hassner, *et al.*, "Learning pose-aware models for pose-invariant face recognition in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 379–393, 2019.

[232] J. Zhao, L. Xiong, J. Li, J. Xing, S. Yan, and J. Feng, "3D-aided dual-agent GANs for unconstrained face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[233] A. Kortylewski, A. Schneider, T. Gerig, B. Egger, A. Morel-Forster, and T. Vetter, "Training deep face recognition systems with synthetic data," *arXiv preprint arXiv:1802.05891*, 2018.

[234] F. Mokhayeri, E. Granger, and G.-A. Bilodeau, "Domain-specific face synthesis for video face recognition from a single sample per person," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 757–772, 2019.

[235] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Deep learning identity-preserving face space," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 113–120, 2013.

[236] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Multi-view perceptron: a deep model for learning face identity and view representations," in *Advances in Neural Information Processing Systems*, pp. 217–225, 2014.

[237] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li, "High-fidelity pose and expression normalization for face recognition in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 787–796, 2015.

[238] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim, "Rotating your face using multi-task deep neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 676–684, 2015.

[239] V. Blanz and T. Vetter, "Face recognition based on fitting a 3D morphable model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1063–1074, 2003.

[240] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the International Conference on Learning Representations*, 2018. Supplemental material: `https://github.com/tkarras/progressive_growing_of_gans`.

[241] Z. Lu, Z. Li, J. Cao, R. He, and Z. Sun, "Recent progress of face image synthesis," in *Proceedings of the 4th IAPR Asian Conference on Pattern Recognition*, pp. 7–12, IEEE, 2017.

[242] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[243] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[244] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[245] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

[246] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[247] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the International Conference on Machine Learning*, pp. 2642–2651, 2017.

[248] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv preprint arXiv:1606.01583*, 2016.

[249] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

[250] T. M. Cover and J. A. Thomas, *Elements of information theory.* John Wiley & Sons, 2012.

[251] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," in *Proceedings of the International Conference on Learning Representations*, 2016.

[252] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain, "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*, pp. 1931–1939, 2015.