Face Image Analysis With Convolutional Neural Networks

Dissertation

Zur Erlangung des Doktorgrades
der Fakultät für Angewandte Wissenschaften
an der Albert-Ludwigs-Universität Freiburg im Breisgau

von

Stefan Duffner

2007

Dekan:	Prof. Dr. Bernhard Nebel
Prüfungskommission:	Prof. Dr. Peter Thiemann (Vorsitz) Prof. Dr. Matthias Teschner (Beisitz) Prof. Dr. Hans Burkhardt (Betreuer) Prof. Dr. Thomas Vetter (Prüfer)
Datum der Disputation:	28. März 2008

Acknowledgments

First of all, I would like to thank Dr. Christophe Garcia for his guidance and support over the last three years. This work would not have been possible without his excellent scientific as well as human qualities and the enormous amount of time he spent for me.

I also want to express my gratitude to my supervisor Prof. Dr. Hans Burkhardt who accompanied me during my thesis, gave me helpful advice and who always welcomed me in Freiburg.

Further, I would like to thank all my colleagues at France Telecom R&D (now Orange Labs), Rennes (France) where I spent three very pleasant years. Notably, Franck Mamalet, Sébastien Roux, Patrick Lechat, Sid-Ahmed Berrani, Zohra Saidane, Muriel Visani, Antoine Lehuger, Grégoire Lefebvre, Manolis Delakis and Christine Barbot.

Finally, I want to say thank you to my parents for their continuing support in every respect.

Abstract

In this work, we present the problem of automatic appearance-based facial analysis with machine learning techniques and describe common specific sub-problems like face detection, facial feature detection and face recognition which are the crucial parts of many applications in the context of indexation, surveillance, access-control or human-computer interaction.

To tackle this problem, we particularly focus on a technique called *Convolutional Neural Network* (CNN) which is inspired by biological evidence found in the visual cortex of mammalian brains and which has already been applied to many different classification problems. Existing CNN-based methods, like the face detection system proposed by Garcia and Delakis, show that this can be a very effective, efficient and robust approach to non-linear image processing tasks such as facial analysis.

An important step in many automatic facial analysis applications, *e.g.* face recognition, is *face alignment* which tries to translate, scale and rotate the face image such that specific facial features are roughly at predefined positions in the image. We propose an efficient approach to this problem using CNNs and experimentally show its very good performance on difficult test images.

We further present a CNN-based method for automatic *facial feature detection*. The proposed system employs a hierarchical procedure which first roughly localizes the eyes, the nose and the mouth and then refines the result by detecting 10 different facial feature points. The detection rate of this method is 96% for the AR database and 87% for the BioID database tolerating an error of 10% of the inter-ocular distance.

Finally, we propose a novel face recognition approach based on a specific CNN architecture learning a non-linear mapping of the image space into a lower-dimensional sub-space where the different classes are more easily separable. We applied this method to several public face databases and obtained better recognition rates than with classical face recognition approaches based on PCA or LDA. Moreover, the proposed system is particularly robust to noise and partial occlusions.

We also present a CNN-based method for the binary classification problem of gender recognition with face images and achieve a state-of-the-art accuracy.

The results presented in this work show that CNNs perform very well on various facial image processing tasks, such as face alignment, facial feature detection and face recognition and clearly demonstrate that the CNN technique is a versatile, efficient and robust approach for facial image analysis.

Zusammenfassung

In dieser Arbeit stellen wir das Problem der automatischen, erscheinungsbasierten Gesichts-Analyse dar und beschreiben gängige, spezifische Unterprobleme wie z.B. Gesichts- und Gesichtsmerkmals-Lokalisierung oder Gesichtserkennung, welche grundlegende Bestandteile vieler Anwendungen im Bereich Indexierung, Überwachung, Zugangskontrolle oder Mensch-Maschine-Interaktion sind.

Um dieses Problem anzugehen, konzentrieren wir uns auf einen bestimmten Ansatz, genannt Neuronales Faltungs-Netzwerk, englisch *Convolutional Neural Network* (CNN), welcher auf biologischen Befunden, die im visuellen Kortex von Säugetierhirnen entdeckt wurden, beruht und welcher bereits auf viele Klassifizierungsprobleme angewandt wurde. Bestehende CNN-basierte Methoden, wie das Gesichts-Lokalisierungs-System von Garcia und Delakis, zeigen, dass dies ein sehr effektiver, effizienter und robuster Ansatz für nicht-lineare Bildverarbeitungs-Aufgaben wie Gesichts-Analyse sein kann.

Ein wichtiger Schritt in vielen Anwendungen der automatischen Gesichts-Analyse, z.B. Gesichtserkennung, ist die *Gesichts-Ausrichtung und -Zentrierung*. Diese versucht das Gesichts-Bild so zu verschieben, zu drehen und zu vergrößern bzw. verkleinern, dass sich bestimmte Gesichtsmerkmale an vordefinierten Bild-Positionen befinden. Wir stellen einen effizienten Ansatz für dieses Problem vor, der auf CNNs beruht, und zeigen experimentell und anhand schwieriger Testbilder die sehr gute Leistungsfähigkeit des Systems.

Darüberhinaus stellen wir eine CNN-basierte Methode zur automatischen *Gesichtsmerkmals-Lokalisierung* vor. Das System bedient sich einem hierarchischen Verfahren, das zuerst grob die Augen, die Nase und den Mund lokalisiert, und dann das Ergebnis verfeinert indem es 10 verschiedene Gesichtsmerkmals-Punkte erkennt. Die Erkennungsrate dieser Methode liegt bei 96% für die AR-Datenbank und 87% für die BioID-Datenbank mit einer Fehler-Toleranz von 10% des Augenabstandes.

Schließlich stellen wir einen neuen Gesichtserkennungs-Ansatz vor, welcher auf einer spezifischen CNN-Architektur beruht und welcher eine nicht-lineare Abbildung vom Bildraum in einen niedrig-dimensionalen Unterraum lernt, in dem die verschiedenen Klassen leichter trennbar sind. Diese Methode wurde auf verschiedene öffentliche Gesichts-Datenbanken angewandt und erzielte bessere Erkennungsraten als klassische Gesichtserkennungs-Ansätze, die auf PCA oder LDA beruhen. Darüberhinaus ist das System besonders robust bezüglich Rauschen und partiellen Verdeckungen.

Wir stellen ferner eine CNN-basierte Methode zum binären Klassifizierung-Problem der *Geschlechtserkennung* mittels Gesichts-Bildern vor und erzielen eine Genauigkeit, die dem aktuellen Stand der Technik entspricht.

Die Ergebnisse, die in dieser Arbeit dargestellt sind, beweisen, dass CNNs sehr gute Leistungen in verschiedenen Gesichts-Bildverarbeitungs-Aufgaben erzielen, wie z.B. Gesichts-Ausrichtung, Gesichtsmerkmals-Lokalisierung und Gesichtserkennung. Sie zeigen außerdem deutlich, dass CNNs ein vielseitiges, effizientes und robustes Verfahren zur Gesichts-Analyse sind.

Résumé

Dans cette thèse, nous proposons le problème de l'analyse faciale basée sur l'apparence avec des techniques d'apprentissage automatique et nous décrivons des sous-problèmes spécifiques tels que la détection de visage, la détection de caractéristiques faciales et la reconnaissance de visage qui sont des composants indispensables dans de nombreuses applications dans le contexte de l'indexation, la surveillance, le contrôle d'accès et l'interaction homme-machine.

Afin d'aborder ce problème, nous nous concentrons sur une technique nommée *réseau de neurones à convolution*, en anglais Convolutional Neural Network (CNN), qui est inspirée des découvertes biologiques dans le cortex visuel des mammifères et qui a déjà été appliquée à de nombreux problèmes de classification. Des méthodes existantes, comme le système de détection de visage proposé par Garcia et Delakis, montrent que cela peut être une approche très efficace et robuste pour des applications de traitement non-linéaire d'images tel que l'analyse faciale.

Une étape importante dans beaucoup d'applications d'analyse facial, comme la reconnaissance de visage, constitue le *recadrage automatique de visage*. Cette technique cherche à décaler, tourner et agrandir ou reduire l'image de visage de sorte que des caractéristiques faciales se trouvent environ à des positions définies préalablement dans l'image. Nous proposons une approche efficace pour ce problème en utilisant des CNNs et nous montrons une très bonne performance de cette approche sur des images de test difficiles.

Nous présentons également une méthode basée CNN pour la détection de caractéristiques faciales. Le système proposé utilise une procedure hiérarchique qui localise d'abord les yeux, le nez et la bouche pour ensuite affiner le résultat en détectant 10 points de caractéristiques faciales différentes. Le taux de détection est de 96 % pour la base AR et de 87 % pour la base BioID avec une tolérance d'erreur de 10 % de la distance inter-oculaire.

Enfin, nous proposons une nouvelle approche de reconnaissance de visage basée sur une architecture spécifique de CNN qui apprend une projection non-linéaire de l'espace de l'image dans un espace de dimension réduite où les classes différentes sont séparables plus facilement. Nous appliquons cette méthode à plusieurs bases publiques de visage et nous obtenons des taux de reconnaissance meilleurs qu'en utilisant des approches classiques basées sur l'Analyse en Composantes Principales (ACP) ou l'Analyse Discriminante Linéaire (ADL). En outre, le système proposé est particulièrement robuste par rapport au bruit et aux occultations partielles.

Nous présentons également une methode basée CNN pour le problème de reconnaissance de genre à partir d'images de visage et nous obtenons un taux comparable à l'état de l'art.

Les résultats présentés dans cette thèse montrent que les CNNs sont très performants dans de nombreuses applications de traitement d'images faciales telles que le recadrage de visage, la détection de caractéristiques faciales et la reconnaissance de visage. Ils démontrent également que la technique de CNN est une approche très variée, efficace et robuste pour l'analyse automatique d'image faciale.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Context

The automatic processing of images to extract semantic content is a task that has gained a lot of importance during the last years due to the constantly increasing number of digital photographs on the Internet or being stored on personal home computers. The need to organize them automatically in a intelligent way using indexing and image retrieval techniques requires effective and efficient image analysis and pattern recognition algorithms that are capable to extract relevant semantic information.

Especially *faces* contain a great deal of valuable information compared to other objects or visual items in images. For example, recognizing a person on a photograph, in general, tells a lot about the overall content of the picture.

In the context of human-computer interaction (HCI), it might also be important to detect the position of specific facial characteristics or recognize facial expressions, in order to allow, for example, a more intuitive communication between the device and the user or to efficiently encode and transmit facial images coming from a camera. Thus, the automatic analysis of face images is crucial for many applications involving visual content retrieval or extraction.

The principal aim of facial analysis is to extract valuable information from face images, such as its position in the image, facial characteristics, facial expressions, the person's gender or identity.

We will outline the most important existing approaches to facial image analysis and present novel methods based on Convolutional Neural Networks (CNN) to detect, normalize and recognize faces and facial features. CNNs show to be a powerful and flexible feature extraction and classification technique which has been successfully applied in other contexts, *i.e.* hand-written character recognition, and which is very appropriate for face analysis problems as we will experimentally show in this work.

We will focus on the processing of two-dimensional gray-level images as this is the most widespread form of digital images and thus allows the proposed approaches to be applied in the most extensive and generic way. However, many techniques described in this work could also be extended to color images, 3D data or multi-modal data.

1.2 Applications

There are numerous possible applications for facial image processing algorithms. The most important of them concern face recognition. In this regard, one has to differentiate between *closed world* and *open world* settings. In a closed world application, the algorithm is dedicated to a limited group of persons, *e.g.* to recognize the members of a family. In an open world context the algorithm should be able to deal with images from "unknown" persons, *i.e.* persons that have not been presented to the system during its design or training. For example, an application indexing large image databases like Google images or television programs should recognize learned persons and respond with "unknown" if the person is not in the database of registered persons.

Concerning face recognition, there further exist two types of problems: *face identification* and *face verification* (or *authentication*). The first problem, face identification, is to determine the identity of a person on an image. The second one only deals with the question: "Is 'X' the identity of the person shown on the image?" or "Is the person shown on the image the one he claims to be?". These questions only require "yes" or "no" as the answer.

Possible applications for face authentication are mainly concerned with access control, *e.g.* restricting the physical access to a building, such as a corporate building, a secured zone of an airport, a house etc. Instead of opening a door by a key or a code, the respective person would communicate an identifier, *e.g.* his/her name, and present his/her face to a camera. The face authentication system would then verify the identity of the person and grant or refuse the access accordingly. This principle could equally be applied to the access to systems, automatic teller machines, mobile phones, Internet sites etc. where one would present his face to a camera instead of entering an identification number or password.

Clearly, also face *identification* can be used for controlling access. In this case the person only has to present his/her face to the camera without claiming his/her identity. A system recognizing the identity of a person can further be employed to control more specifically the rights of the respective persons stored in its database. For instance, parents could allow their children to watch only certain television programs or web sites, while the television or computer would automatically recognize the persons in front of it.

Video surveillance is another application of face identification. The aim here is to recognize suspects or criminals using video cameras installed at public places, such as banks or airports, in order to increase the overall security of these places. In this context, the database of suspects to recognize is often very large and the images captured by the camera are of low quality, which makes the task rather difficult.

With the vast propagation of digital cameras in the last years the number of digital images stored on servers and personal home computers is rapidly growing. Consequently, there is an increasing need of indexation systems that automatically categorize and annotate this huge amount of images in order to allow effective searching and so-called content-based image retrieval. Here, face detection and recognition methods play a crucial role because a great part of photographs actually contain faces. A similar application is the temporal segmentation and indexation of video sequences, such as TV programs, where different scenes are often characterized by different faces.

Figure 1.1: An example face under a fixed view and varying illumination

Another field of application is facial image compression, *i.e.* parts of images containing faces can be coded by a specialized algorithm that incorporates a generic face model and thus leads to very high compression rates compared to universal techniques.

Finally, there are many possible applications in the field of advanced Human-Computer Interaction (HCI), *e.g.* the control and animation of avatars, *i.e.* computer synthesized characters. Such systems capture the position and movement of the face and facial features and accordingly animate a virtual avatar, which can be seen by the interlocutor. Another example would be the facilitation of the interaction of disabled persons with computers or other machines or the automatic recognition of facial expressions in order to detect the reaction of the person(s) sitting in front of a camera (*e.g.* smiling, laughing, yawning, sleeping).

1.3 Difficulties

There are some inherent properties of faces as well as the way the images are captured which make the automatic processing of face images a rather difficult task. In the case of face recognition, this leads to the problem that the *intra-class* variance, *i.e.* variations of the face of the same person due to lighting, pose etc., is often higher than the *inter-class* variance, *i.e.* variations of facial appearance of different persons, and thus reduces the recognition rate. In many face analysis applications, the appearance variation resulting from these circumstances can also be considered as *noise* as it makes the desired information, *i.e.* the identity of the person, harder to extract and reduces the overall performance of the respective systems.

In the following, we will outline the most important difficulties encountered in common real-world applications.

1.3.1 Illumination

Changes in illumination can entail considerable variations of the appearance of faces and thus face images. Two main types of light sources influence the overall illumination: ambient light and point light (or directed light). The former is somehow easier to handle because it only affects the overall brightness of the resulting image. The latter however is far more difficult to analyze, as face images taken under varying light source directions follow a highly non-linear function. Additionally, the face can cast shadows on itself. Figure 1.1 illustrates the impact of different illumination on face images.

Figure 1.2: An example face under fixed illumination and varying pose

Figure 1.3: An example face under fixed illumination and pose but varying facial expression

Many approaches have been proposed to deal with this problem. Some face detection or recognition methods try to be invariant to illumination changes by implicitly modeling them or extracting invariant features. Others propose a separate processing step, a kind of normalization, in order to reduce the effect of illumination changes. In section 4.3 some of these illumination normalization methods will be outlined.

1.3.2 Pose

The variation of head pose or, in other words, the viewing angle from which the image of the face was taken is another difficulty and essentially impacts the performance of automatic face analysis methods. For this reason, many applications limit themselves to more or less *frontal* face images or otherwise perform a pose-specific processing that requires a preceding estimation of the pose, like in multi-view face recognition approaches. Section 4.4 outlines some 2D pose estimation approaches that have been presented in the literature.

If the rotation of the head coincides with the image plane the pose can be normalized by estimating the rotation angle and turning the image such that the face is in an upright position. This type of normalization is part of a procedure called *face alignment* or *face registration* and is described in more detail in section 4.5.

Figure 1.2 shows some example face images with varying head pose.

1.3.3 Facial Expressions

The appearance of a face with different facial expressions varies considerably (see Fig. 1.3). Depending on the application, this can be of more or less importance. For example, for access control systems the subjects are often required to show a neutral expression. Thus, invariance to facial expression might not be an issue in this case. On the contrary, in an image or video indexation system, for

example, this would be more important as the persons are shown in every-day situations and might speak, smile, laugh etc.

In general, the mouth is subject to the largest variation. The respective person on an image can have an open or closed mouth, can be speaking, smiling, laughing or even making grimaces.

Eyes and eyebrows are also changing subject to varying facial expressions, *e.g.* when the respective person blinks, sleeps or widely opens his/her eyes.

1.3.4 Partial Occlusions

Partial occlusions occur quite frequently in real-world face images. They can be caused by a hand occluding a part of the face, *e.g.* the mouth, by long hear, glasses, sun glasses or other objects or persons.

In most of the cases, however, the face occludes parts of itself. For example, in a view from the side the other side of the face is hidden. Also, a part of the cheek can be occluded by the nose or an eye can be covered by its orbit for example.

1.3.5 Other types of variations

Appearance variations a also caused by varying make-up, varying hair-cut and the presence of facial hear (beard, mustache etc.).

Varying age is also an important factor influencing the performance of many face analysis methods. This is the case for example in face recognition when the reference face image has been taken some years before the image to recognize.

Finally, there are also variations across the subjects' identities, such as race, skin color or, more generally, ethnic origin. The respective differences in the appearance of the face images can cause difficulties in applications like face or facial feature detection or gender recognition.

1.4 Objectives

The goals pursued in this work principally concern the evaluation of Convolutional Neural Networks (CNN) in the context of facial analysis applications. More specifically, we will focus on the following objectives:

- evaluate the performance of CNNs w.r.t. appearance-based facial analysis
- investigate the robustness of CNNs against classical sources of noise in the context of facial analysis
- propose different CNN architectures designed for specific facial analysis problems such as face alignment, facial feature detection, face recognition and gender classification
- improve upon the state-of-the-art in appearance-based facial feature detection, face alignment as well as face recognition under real-world conditions
- investigate different solutions improving the performance of automatic face recognition systems

1.5 Outline

In the following chapter we will outline some of the most important machine learning techniques used for object detection and recognition in images, such as statistical projection methods, Hidden Markov Models, Support Vector Machines and Neural Networks.

In chapter 3, we will then focus on one particular approach, called Convolutional Neural Networks (CNN), which is the foundation for the methods proposed in this work.

Having described, among other aspects, the principle architecture and training methods for CNNs, in chapter 4 we will outline the problem of face detection and normalization and how CNNs can tackle these types of problems. Using an existing CNN-based face detection system, called Convolutional Face Finder (CFF), we will further present an effective approach for *face alignment* which is an important step in many facial analysis applications.

In chapter 5, we will describe the problem of *facial feature detection* which shows to be crucial for any facial image processing task. We will propose an approach based on a specific type of CNN to solve this problem and experimentally show its performance in terms of precision and robustness to noise.

Chapter 6 outlines two further facial analysis problems, namely *automatic face recognition* and *gender recognition*. We will also present CNN-based approaches to these problems and experimentally show their effectiveness compared to other machine learning techniques proposed in the literature.

Finally, chapter 7 will conclude this work with a short summary and some perspectives for future research.

Chapter 2

Machine Learning Techniques for Object Detection and Recognition

2.1 Introduction

In this chapter we will outline some of the most common machine learning approaches to object detection and recognition. Machine Learning techniques automatically learn from a set of examples how to classify new instances of the same type of data. The capacity to generalize, *i.e.* the ability to successfully classify unknown data and possibly infer generic rules or functions, is an important property of these approaches and is sought to be maximized.

Usually, one distinguishes between three types of learning:

Supervised learning A training set *and* the corresponding desired outputs of the function to learn are available. Thus, during training the algorithm iteratively presents examples to the system and adapts its parameters according to the distance between the produced and the desired outputs.

Unsupervised learning The underlying structure of the training data, *i.e.* the desired output, is unknown and is to be determined by the training algorithm. For example, for a classification method this means that the class information is not available and has to be approximated by grouping the training examples using some distance measure, a technique called *clustering*.

Reinforcement learning Here, the exact output of the function to learn is unknown, and training consists in a parameter adjustment based on only two concepts, reward and penalty. That is, if the system does not perform well (enough) it is "penalized" and the parameters are adapted accordingly. Otherwise, it is "rewarded", *i.e.* some positive reinforcement takes place.

Most of the algorithms described in the following are supervised, but they are employed for rather different purposes: some of them are used to extract

features from the input data, some are used to classify the extracted features, and others perform both tasks.

The application context varies also largely, *i.e.* some of the approaches can be used for *detection* of features and/or objects, some only for *recognition* and others for both. Further, in many systems a combination of several of the techniques described in this chapter is used. Thus, in a sense, they could be considered as some kind of *building blocks* for effective object detection and recognition systems.

Let us begin with some of the most universal techniques used in machine learning which are based on a statistical analysis of the data allowing to significantly reduce its dimensionality and extract valuable information.

2.2 Statistical Projection Methods

In order to be able to automatically analyze images, they are often resized to have a certain width w and height h. Then, the respective image rows or columns of each image are concatenated to build a vector of dimension $n = w \times h$. The resulting vector space is called *image space*, denoted $\mathcal{I}$ in the following.

In signal processing tasks there is often a lot of redundancy in the respective images/vectors because, firstly, images of the same class of objects are likely to be similar and, secondly, neighboring pixels in an image are highly correlated.

Thus, it seems obvious to represent the images in a more compact form, *i.e.* to project the vectors into a subspace $\mathcal{S}$ of $\mathcal{I}$ by means of a statistical projection method. In the literature, the terms *dimensionality reduction* or *feature selection* are often employed in the context of these techniques. These methods aim at computing $\mathcal{S}$ which, in general, is of lower dimension than $\mathcal{I}$, such that the transformed image vectors are statistically less correlated. There are two main groups of projections: *linear* and *non-linear* projections.

Linear projection techniques transform an image vector $\mathbf{x} = (x_1, \ldots, x_n)^T$, of dimension n into a vector $\mathbf{s} = (s_1, \ldots, s_k)^T$ of dimension k, by a linear $k \times n$ transformation matrix W:

$$\mathbf{s} = W^T \mathbf{x} \tag{2.1}$$

In general, one eliminates those basis vectors that are supposed to contain the least important information for a given application using a predefined criteria. Thus, the dimension k of the resulting subspace $\mathcal{S}$ can be chosen after calculating the basis vectors spanning the entire subspace.

The most common and fundamental projection methods are the *Principal Component Analysis* (PCA) and the *Linear Discriminant Analysis* (LDA) which will be described in the following sections.

Non-linear approaches are applied when a linear projection does not suffice to represent the data in a way that allows the extraction of discriminant features. This is the case for more complex distributions where mere hyperplanes fail to separate the classes to distinguish. As most of these approaches are iterative, they require an *a priori* choice of the dimension k of the resulting subspace $\mathcal{S}$.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA), also known as the discrete Karhunen-Loève Transform (KLT) or Hotelling Transform as it is due to Hotelling [101], is a linear orthogonal projection into the subspace where the first dimension (or axis) corresponds to the direction of $\mathcal{I}$ having the greatest variance, the second dimension to the direction with the second greatest variance and so on.

Thus, the resulting orthogonal subspace $\mathcal{S}$, called principal subspace, describes best the distribution of the input space $\mathcal{I}$. It finds the directions of greatest variance, which are supposed to reflect the most "important" aspects of the data.

Given a certain number N of input vectors $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$ ($\mathbf{x_i} \in \mathbb{R}^n$) that are assumed to have a multi-normal distribution and to be centered, *i.e.* $\frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i} = 0$, the corresponding projected vectors are

$$\mathbf{s_i} = W^T \mathbf{x_i} \qquad i \in 1..N \quad , \tag{2.2}$$

where $\mathbf{s_i} \in \mathbb{R}^k$. Now let Σ be the covariance matrix of the input vectors

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i}\ \mathbf{x_i}^T \quad . \tag{2.3}$$

Hence, the covariance matrix of the projected vectors $\mathbf{s_i}$ is defined as

$$\Sigma' = W^T \Sigma W \quad . \tag{2.4}$$

Finally, the projection matrix W is supposed to maximize the variance of the projected vectors. Thus,

$$W = \underset{\tilde{W}}{\operatorname{argmax}} |\tilde{W}^T \Sigma \tilde{W}| \quad . \tag{2.5}$$

The k columns of W, *i.e.* the basis vectors of $\mathcal{S}$, are called the principal components and represent the *eigenvectors* corresponding to the largest eigenvalues of the covariance matrix Σ.

An important characteristic of PCA is that if $k < n$ the reconstruction error e in terms of the Euclidean distance is minimal,

$$e = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{x_i} - W\,\mathbf{s_i} \right\| \quad . \tag{2.6}$$

Thus, the first k eigenvectors form a subspace that optimally encodes or represents the input space $\mathcal{I}$. This fact is exploited for example in compression algorithms and template matching techniques.

The choice of k depends largely upon the actual application. Additionally, for some applications it might not even be optimal to select the eigenvectors corresponding to the largest eigenvalues.

Kirby *et al.* [122] introduced a classical selection criteria which they call *energy dimension*. Let λ_j be the eigenvalue associated with the j^{th} eigenvector. Then, the energy dimension of the i^{th} eigenvector is:

$$E_i = \frac{\sum_{j=i+1}^{n} \lambda_j}{\sum_{j=1}^{n} \lambda_j} \quad . \tag{2.7}$$

One can show that the Mean Squared Error (MSE) produced by the last $n-i$ rejected eigenvectors is $\sum_{j=i+1}^{n} \lambda_j$. The selection of k now consists in determining a threshold τ such that $E_{k-1} > \tau$ and $E_k < \tau$.

Apart from image compression and template matching, PCA is often applied to classification tasks, *e.g.* the *Eigenfaces* approach [243] in face recognition. Here, the projected vectors $\mathbf{s_i}$ are the signatures to be classified. To this end, the signatures of the N input images are each associated with a class label and used to build a classifier. The most simple classifier would be a nearest neighbor classifier using an Euclidean distance measure.

To sum up, PCA calculates the linear orthogonal subspace having its axes oriented with the directions of greatest variances. It thus optimally represents the input data. However, in a classification context it is not guaranteed that in the subspace calculated by PCA the separability of the data is improved. In this regard, the Linear Discriminant Analysis (LDA) described in the following section is more suitable.

2.2.2 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) has been introduced by Fisher [69] in 1936 but generalized later on to the so-called Fisher's Linear Discriminant (FLD). It is, in contrast to the PCA, not only concerned with the best representation of the data but also with its separability in the projected subspace with regard to the different classes.

Let $\Omega = \{\mathbf{x_1}, \dots \mathbf{x_N}\}$ be the training set partitioned into c annotated classes denoted Ω_i ($i \in 1..c$). We are now searching the subspace $\mathcal{S}$ that maximizes the inter-class variability while minimizing the intra-class variability, thus improving the separability of the respective classes. To this end, one maximizes the so-called Fisher's criterion [69, 16]:

$$J(W) = \frac{|W^T \Sigma_b W|}{|W^T \Sigma_w W|} \quad . \tag{2.8}$$

Thus,

$$W = \underset{\tilde{W}}{\operatorname{argmax}} \frac{|\tilde{W}^T \Sigma_b \tilde{W}|}{|\tilde{W}^T \Sigma_w \tilde{W}|} \quad , \tag{2.9}$$

where

$$\Sigma_w = \frac{1}{N} \sum_{j=1}^{c} \sum_{\mathbf{x_i} \in \Omega_j} (\mathbf{x_i} - \overline{\mathbf{x_j}})(\mathbf{x_i} - \overline{\mathbf{x_j}})^T \tag{2.10}$$

represents the within-class variance and

$$\Sigma_b = \frac{1}{N} \sum_{j=1}^{c} N_j (\overline{\mathbf{x_j}} - \overline{\mathbf{x}})(\overline{\mathbf{x_j}} - \overline{\mathbf{x}})^T \tag{2.11}$$

the between-class variance. N_j is the number of examples in Ω_j (*i.e.* of class j) and $\overline{\mathbf{x_j}}$ are the respective means, *i.e.* $\overline{\mathbf{x_j}} = \frac{1}{N_j} \sum_{\mathbf{x_i} \in \Omega_j} \mathbf{x_i}$. $\overline{\mathbf{x}}$ is the overall mean of the data which is assumed to be centered, *i.e.* $\overline{\mathbf{x}} = 0$.

The projection matrix W is obtained by calculating the eigenvectors associated with the largest eigenvalues of the matrix $\Sigma_w^{-1}\Sigma_b$. These eigenvectors form the columns of W.

A problem occurs when the number of examples N is smaller than the size of the input vectors, *i.e.* for images the number of pixels n. Then, Σ_w is singular since its rank is at most $N-c$. The calculation of Σ_w^{-1} is thus impossible. Several approaches have been proposed to overcome this problem. One is to produce additional examples by adding noise to the images of the training database. Another approach consists in first applying PCA to reduce the input vector space to the dimension $N - c$ and then perform LDA as described above.

2.2.3 Other Projection Methods

There are many other projection techniques proposed in the literature and which can possibly be applied to object detection and recognition.

For example, Independent Component Analysis (ICA) [17, 2, 35, 109, 108] is a technique often used for blind source separation [120], *i.e.* to find the different independent sources a given signal is composed of. ICA seeks a linear sub-space where the data is not only uncorrelated but *statistically independent*. In its most simple form, the model is the following:

$$\mathbf{x} = \mathbf{A}^T \mathbf{s} \,, \tag{2.12}$$

where $\mathbf{x}$ is the observed data, $\mathbf{s}$ are the independent sources and $\mathbf{A}$ is the so-called mixing matrix. ICA consists in optimizing an objective function, denoted contrast function, that can be based on different criteria. The contrast function has to ensure that the projected data is independent and non-Gaussian. Note that ICA does not reduce the dimensionality of the input data. Hence, it is often employed in combination with PCA or any other dimensionality reduction technique. Numerous implementations of ICA exist, *e.g.* INFOMAX [17], JADE [35] or FastICA [109].

Yang *et al.* [263] introduced the so-called two-dimensional PCA, which does not require the input image to be transformed into a one-dimensional vector beforehand. Instead, a generalized covariance matrix is directly estimated using the image matrices. Then, the eigenvectors are determined in a similar manner than for 1D-PCA by minimizing a special criterion based on this covariance matrix. Finally, in order to perform classification a distance measure between matrix signatures has to be defined. It has been shown that this method outperforms one-dimensional PCA in terms of classification rate [263] and robustness [254].

Visani *et al.* [252] presented a similar approach based on LDA: the two-dimensional oriented LDA. The procedure is analogical to the 2D-PCA method where the projection is directly performed on the image matrices, either column-wise or row-wise. A generalized Fisher's criterion is defined and minimized in order to obtain the projection matrix. Further, the authors showed that in contrast to LDA, the two-dimensional oriented LDA can implicitly circumvent the singularity problem. In a later work [253], they generalized this approach to the Bilinear Discriminant Analysis (BDA) where column-wise and row-wise 2D-LDA is iteratively applied to estimate the pair of projection matrices minimizing an expression similar to the Fisher's criterion which combines the two projections.

Note that the projection methods presented so far are all *linear* projection techniques. However, in some cases the different classes cannot be correctly

separated in a linear sub-space. Then, *non-linear* projection methods can help to improve the classification rate. Most of the linear projection methods can be made non-linear by projecting the input data into a higher-dimensional space where the classes are more likely to be linearly separable. That means, the separating hyperplane in this sub-space represents a non-linear sub-space of the input vector space. Fortunately, it is not necessary to explicitly describe this higher-dimensional space and the respective projection function if we find a so-called *kernel function* that implements a simple dot-product in this vector space and satisfies the Mercer's condition (see Theorem 1 on p. 22). For a more formal explanation see section 2.6.3 on non-linear SVMs. The kernel function allows to perform a dot-product in the target vector space and can be used to construct non-linear versions of the previously described projection techniques *e.g.* PCA [219, 264], LDA [161] or ICA [6].

The projection approaches that have been outlined in this section can in principal be applied to any type of data in order to perform a statistical analysis on the respective examples. A technique called Active Appearance Model (AAM) [41] can also be classified as a statistical projection approach but it is much more specialized to model images of deformable objects under varying external conditions. Thus, in contrast to methods like PCA or LDA, where the input image is treated as a "static" vector, small local deformations are taken into account. AAMs have been especially applied to face analysis, and we will therefore describe this technique in more detail in the following section.

2.3 Active Appearance Models

Active Appearance Models (AAM), introduced by Cootes *et al.* [41] as an extension to Active Shape Models (ASM) [43], represent an approach that statistically describes not only the texture of an object but also its shape. Given a new image of the class of objects to analyze, the idea is here to interpret the object by synthesizing an image of the respective object while approximating as good as possible its appearance in the real image. It has mainly been applied to face analysis problems [60, 41]. Therefore, face images we will used in the following to illustrated this technique. Modeling the shape of faces appears to be helpful in most face analysis applications where the face images are subject to changes in pose and facial expressions.

2.3.1 Modeling shape and appearance

The basis of the algorithm is a set of training images with a certain number of annotated feature points, so-called landmark points, *i.e.* two-dimensional vectors. Each set of landmarks is represented as a single vector $\mathbf{x}$, and PCA is applied to the whole set of vectors. Thus any shape example can be approximated by the equation:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{P_s}\mathbf{b_s} \quad , \tag{2.13}$$

where $\mathbf{x}$ is the mean shape, and $\mathbf{P_s}$ is the linear subspace representing the possible variations of shape parameterized by the vector $\mathbf{b_s}$.

Then, the annotated control points of each training example are matched to the mean shape while warping the pixel intensities using a triangulation algorithm. This leads to a so-called *shape-free* face patch for each example.

Figure 2.1: Active Appearance Models: annotated training example and corresponding shape-free patch

Figure 2.1 illustrates this with an example face image. Subsequently, a PCA is performed on the gray values $\mathbf{g}$ of the shape-free images forming a statistical model of texture:

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{P_g b_g} \quad , \tag{2.14}$$

where $\overline{\mathbf{g}}$ represents the mean texture, and the matrix $\mathbf{P_g}$ linearly describes the texture variations parameterized by the vector $\mathbf{b_g}$.

Since shape and texture are correlated, another PCA is applied on the concatenated vectors of $\mathbf{b_s}$ and $\mathbf{b_g}$ leading to the combined model:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{Q_s c} \tag{2.15}$$

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{Q_g c} \quad , \tag{2.16}$$

where c is a parameter controlling the overall appearance, *i.e.* both shape and texture, and $\mathbf{Q_s}$ and $\mathbf{Q_g}$ represent the combined linear shape-texture subspace.

Given a parameter vector c, the respective face can be synthesized by first building the shape-free image, *i.e.* the texture, using equation 2.16 and then warping the face image by applying equation 2.15 and the triangulation algorithm used to build the shape-free patches.

2.3.2 Matching the model

Having built the statistical shape and texture models, the objective is to match the model to an image by synthesizing the approximate appearance of the object in the real image. Thus, we want to minimize:

$$\Delta = |\mathbf{I_i} - \mathbf{I_m}| \quad , \tag{2.17}$$

where $\mathbf{I_i}$ is the vector of gray-values of the real image and $\mathbf{I_m}$ is the one of the synthesized image.

The approach assumes that the object is roughly localized in the input image, *i.e.* during the matching process, the model with its landmark points must not be too far away from the resulting locations.

Now, the decisive question is how to change the model parameters c in order to minimize Δ. A good approximation appears to be a linear model:

$$\delta\mathbf{c} = \mathbf{A}(\mathbf{I_i} - \mathbf{I_m}) \quad , \tag{2.18}$$

where $\mathbf{A}$ is determined by a multi-variate linear regression on the training data augmented by examples with manually added perturbations.

To calculate $\mathbf{I_i} - \mathbf{I_m}$, the respective real and synthesized images are transformed to be shape-free using a preliminary estimate of the shape model. Thus, we compute:

$$\delta\mathbf{g} = \mathbf{g_i} - \mathbf{g_m} \tag{2.19}$$

and obtain

$$\delta\mathbf{c} = \mathbf{A}\delta\mathbf{g} \quad . \tag{2.20}$$

This linear approximation shows to perform well over a limited range of the model parameters, *i.e.* about 0.5 standard deviations of the training data.

Finally, this estimation is put into an iterative framework, *i.e.* at each iteration we calculate:

$$\mathbf{c}' = \mathbf{c} - \mathbf{A}\delta\mathbf{g} \tag{2.21}$$

until convergence, where the matrix $\mathbf{A}$ is scaled such that it minimizes $|\delta\mathbf{g}|$.

The final result can then be used, for example, to localize specific feature points, to estimate the 3D orientation of the object, to generate a compressed representation of the image or, in the context of face analysis, to identify the respective person, gender or facial expression.

Clearly, AAMs can cope with small local image transformations and elegantly model shape and texture of an object based on a preceding statistical analysis of the training examples. However, the resulting projection space can be rather large, and the search in this space, *i.e.* the matching process, can be slow. A fundamentally different approach to take into account local transformations of a signal are Hidden Markov Models (HMM). This is a probabilistic method that represents a signal, *e.g.* an image, as a *sequence* of observations. The following section outlines this approach.

2.4 Hidden Markov Models

2.4.1 Introduction

Hidden Markov Models (HMM), introduced by Rabiner *et al.* [190, 191], are commonly used to model the sequential aspect of data. In the signal processing context for example, they have been frequently applied to speech recognition problems modeling the temporal sequence of states and observations, *e.g.* phonemes. An image can also be seen as a sequence of observations, *e.g.* image subregions, and here the image either has to be linearized into a one-dimensional structure or special types of HMMs have to be used, for example two-dimensional Pseudo HMMs or Markov Random Fields.

Being the most common approaches in image analysis, we will focus on 1D and Pseudo 2D HMMs in the following. The major disadvantage of "real" 2D HMMs is their relatively high complexity in terms of computation time.

A HMM is characterized by a finite number of states, and it can be in only one state at a time (as a finite state machine). The *initial state probabilities* define, for every state, the probability of the HMM being in that state at time $t = 1$. For each following time step $t = 2..T$ it can either change the state or stay in the same state with a certain probability defined by the so-called *transition probabilities*. Further, in any state it creates an output from a pre-defined

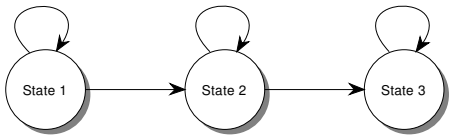


Figure 2.2: A left-right Hidden Markov Model

vocabulary with a certain probability, determined by the *output probabilities*. At time $t = T$ the HMM will have produced a certain sequence of outputs, called observations; $O = \{o_1, \ldots, o_T\}$. The sequence of states $Q = \{q_1, \ldots, q_T\}$ it has traversed, however, is unknown (hidden) and has to be estimated by analyzing the observation whereas a single observation could be produced by *different* state sequences.

Fig. 2.2 illustrates a simple example of a HMM with 3 states. This type of HMM is called left-right model or Bakis model.

More formally we can describe a HMM as follows:

Definition 1 *A Hidden Markov Model is defined as* $\lambda = \{S, V, A, B, \Pi\}$*, where*

- $S = \{s_1, \ldots, s_N\}$ *is the set of* N *possible states,*
- $V = \{v_1, \ldots, v_L\}$ *is the set of* L *possible outputs constituting the vocabulary,*
- $A = \{a_{ij}\}_{i,j=1..N}$ *is the set transition probabilities from state* i *to state* j*,*
- $B = \{b_i(l)\}_{i=1..N,l=1..L}$ *define the output probabilities of output* l *in state* i*,*
- $\Pi = \{\pi_1, \ldots, \pi_N\}$ *is the set of initial state probabilities.*

Note that

$$\sum_{i=1}^{N} \pi_i = 1 \quad , \tag{2.22}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i = 1, \ldots, N \quad \text{and} \tag{2.23}$$

$$\sum_{l=1}^{L} b_i(l) = 1 \qquad \forall i = 1, \ldots, N \quad . \tag{2.24}$$

Given a HMM λ, the goal is to determine the probability of a new observation sequence $O = \{o_1, \ldots, o_T\}$, *i.e.* $P[O|\lambda]$. For this purpose, there are several algorithms, the most simple one being explained in the following section.

2.4.2 Finding the most likely state sequence

There are many algorithms for estimating $P[O|\lambda]$ and the most likely state sequence $Q^* = \{q_1^*, \ldots, q_T^*\}$ having generated O. The most well known of these are called *Viterbi algorithm* and *Baum-Welsh algorithm*. Algorithm 1 describes the former which is a kind of simplification of the latter. Note that δ_{ti} denotes

Algorithm 1 The Viterbi algorithm

for $i = 1$ to N **do**
  $\delta_{1i} = \pi_i b_i(o_1)$
end for
for $t = 2$ to T **do**
  for $i = 1$ to N **do**
    $\delta_{ti} = b_i(o_t) \max\{\delta_{t-1,j} a_{ji} \ \ \forall j = 1..N\}$
    $\phi_{ti} = s_j$ where $j = \text{argmax}_j\{\delta_{t-1,j} a_{ji} \ \ \forall j = 1..N\}$
  end for
end for
$P[O|\lambda] = \max\{\delta_{Tj} \ \ \forall j = 1..N\}$
$q_T^* = \text{argmax}_j\{\delta_{Tj} \ \ \forall j = 1..N\}$
for $t = T - 1$ to 1 **do**
  $q_t^* = \phi_{t+1,q_{t+1}^*}$
end for

the probability of being in state s_i at time t, and ϕ_{ti} denotes the most probable preceding state being in s_i at time t. Thus, the ϕ_{ti} store the most probable state sequence. The last loop allows to retrieve the final most likely state sequence Q^* by recursively traversing ϕ_{ti}.

When applying a HMM to a given observation sequence O it suffices for most applications to calculate $P[O|\lambda]$ as stated above. The actual state sequence Q^* however is necessary for the training process explained in the following section.

2.4.3 Training

In order to automatically determine and re-adjust the parameters of λ a set of training observations $O_{tr} = \{o_{t1}, \ldots, o_{tM}\}$ is used, and a training algorithm, for example algorithm 1, is applied to estimate the probabilities: $P[O_{tr}|\lambda]$ and $P[O_{tr}, q_t = s_i|\lambda]$ for every state s_i at every time step t.

Then each parameter can be re-estimated by re-generating the observation sequences O_{tr} and "counting" the number of events determining the respective parameter. For example, to adjust a_{ij} one calculates:

$$a'_{ij} = \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} = \frac{P[q_t = s_i, q_{t+1} = s_j|O_{tr}, \lambda]}{P[q_t = s_i|O_{tr}, \lambda]} \quad (2.25)$$

The output probabilities B and the initial state probabilities Π are estimated in an analogical way. However, the number and topology of states S has to be determined experimentally in most cases.

2.4.4 HMMs for Image Analysis

HMMs are one-dimensional models and have initially been applied to the processing of audio data [190]. However, there are several approaches to adapt this technique to 2D data like images.

(a) 1D HMM based on image bands (b) 1D HMM based on image blocks

Figure 2.3: Two simple approaches to image analysis with 1D HMMs

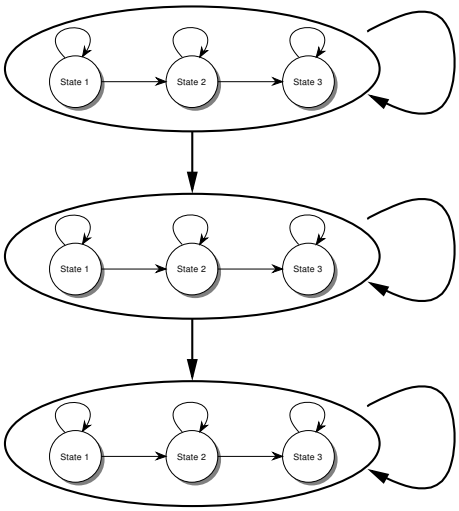


Figure 2.4: Illustration of a 2D Pseudo-HMM

One of them [214] is to consider an image as a sequence of horizontal bands, possibly overlapping and spreading from top to bottom. Fig. 2.3(a) illustrates this. The HMM consequently has a left-right topology. Visual features of the image bands, *e.g.* pixel intensities, then correspond to the outputs of the HMM.

A similar approach is to partition the image into a set of blocks of predefined size. A one-dimensional sequence is then formed by concatenating the lines (or columns) of blocks. Fig. 2.3(b) illustrates this procedure. Additional constraints can be added in order to ensure that certain states correspond to the end of lines in the image.

Finally, an approach called 2D Pseudo-HMM uses a hierarchical concept of super-states, illustrated in Fig. 2.4. The super-states form a vertical 1D sequence corresponding to the lines (or bands) of the image. Each super-state in turn contains a 1D HMM modeling the sequence of horizontal observations (pixels or blocks) in a line. Thus, determining the hidden state sequence Q of an observation O implies a two-level procedure, *i.e.* first, to calculate the most likely sequence of super-states using the lines or bands of the image and, secondly, to determine the most likely sequence of sub-states corresponding to each line independently.

Obviously, HMMs are very suitable for modeling sequential data, and thus

they are principally used in signal processing tasks. Let us now consider some more general machine learning techniques which do not explicitely model this sequential aspect but, on the other hand, can more easily and efficiently be applied to higher dimensional data such as images. Adaptive Boosting is one such approach and will be explained in the following section.

2.5 Adaboost

2.5.1 Introduction

Adaptive Boosting, short *Adaboost*, is a classification technique introduced by Freund and Schapire [70]. The basic idea here is to combine several "weak" classifiers into a single "strong" classifier, where the weak classifiers perform only slightly better than just random guessing.

The principle of the algorithm is to learn a global binary decision function by iteratively adding and training weak classifiers, *e.g.* wavelets networks or Neural Networks, while focusing on more and more difficult examples. It has been applied to many classification problems and has become a widely used machine learning technique due to its simplicity and performance in terms of classification rate and computation time.

2.5.2 Training

Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ be the training set where the $x_i \in X$ are the training examples and $y_i \in Y$ the respective class labels. We will focus here on the basic Adaboost algorithm where $Y = \{-1, +1\}$ but extensions to multi-class classification have been proposed in the literature [71, 216].

The procedure is as follows: at each iteration $t = 1..T$ a weak classifier $h_t : X \to \{-1, +1\}$ is trained using the training examples weighted by a set of weights $D_t(i)$, $i = 1..m$. Then, the weights corresponding to misclassified examples are increased and weights corresponding to correctly classified examples are decreased. Thus, the algorithm focuses more and more on harder examples. The final decision $H(x)$ calculated by the strong classifier is then a weighted sum of the weak decisions $h_t(x)$ where the weights α_t are chosen to be inversely proportional to the error ϵ_t of the classifier h_t, *i.e.* if the error is large the respective classifier will have less influence on the final decision. Algorithm 2 describes the basic Adaboost algorithm. The variable Z_t is a normalization constant in order to make D_{t+1} a distribution.

Now, let $\gamma_t = \frac{1}{2} - \epsilon_t$, *i.e.* the improvement of the classifier over a random guess. It has been proven [71] that the upper bound of the error on the training set is:

$$\prod_t \left[2\sqrt{\epsilon(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right) \quad . \tag{2.26}$$

Thus, if $\gamma_t > 0$, *i.e.* each hypothesis is only slightly better than random, the training error drops exponentially fast.

Schapire *et al.* [215] also conducted theoretical studies in terms of the generalization error. To this end, they define the *margin* of the training examples

Algorithm 2 The Adaboost algorithm

1: $D_1(i) = 1/m \quad \forall i = 1..m$
2: **for** $t = 1$ to T **do**
3: Train weak classifier $h_t(i)$ using the distribution D_t
4: Calculate the produced error:

$$\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

5: Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

7: **end for**
8: Output the final decision function:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

as:

$$\text{margin}(x, y) = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \quad , \tag{2.27}$$

i.e. a value in the interval $[-1, +1]$ and positive if and only if the example is correctly classified. Then, they show that the generalization error is with a high probability upper bounded by:

$$\hat{\Pr}[\text{margin}(x, y) \leq \theta] + \tilde{O} \left(\sqrt{\frac{d}{m\theta^2}} \right) \tag{2.28}$$

for any $\theta > 0$, where $\hat{\Pr}[\cdot]$ denotes the empirical probability on the training set and d the VC-dimension of the weak classifiers.

Adaboost is a very powerful machine learning technique as it can turn any weak classifier into a strong one by linearly combining several instances of it. A completely different classification approach called Support Vector Machine (SVM) is based on the principal of Structural Risk Minimization which not only tries to minimize the classification error on the training examples but also takes into account the ability of the classifier to generalize to new data. The following section explains this approach in more detail.

2.6 Support Vector Machines

2.6.1 Structural Risk Minimization

The classification technique called Support Vector Machine (SVM) [23, 246, 44] is based on the principle of Structural Risk Minimization (SRM) formulated by Vapnik *et al.* [245]. One of the basic ideas of this theory is that the test error

rate, or structural risk $R(\alpha)$, is upper bounded by the training error rate, or empirical risk R_{emp} and an additional term called VC-confidence which depends on the so-called Vapnik-Chervonenkis (VC)-dimension h of the classification function. More precisely, with the probability $1-\eta$, the following holds [246]:

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l}} \quad , \tag{2.29}$$

where α are the parameters of the function to learn and l is the number of training examples. The VC-dimension h of a class of functions describes its "capacity" to classify a set of training data points. For example, in the case of a two-class classification problem, if a function f has a VC-dimension of h there exists at least one set of h data points that can be correctly classified by f, *i.e.* assigned the label -1 or $+1$ to it. If the VC-dimension is too high the learning machine will overfit and show poor generalization. If it is too low, the function will not sufficiently approximate the distribution of the data and the empirical error will be too high. Thus, the goal of SRM is to find a h that minimizes the structural risk $R(\alpha)$, which is supposed to lead to maximum generalization.

2.6.2 Linear Support Vector Machines

Vapnik [246] showed that for linear hyperplane decision functions:

$$f(\mathbf{x}) = \text{sign}((\mathbf{w} \cdot \mathbf{x}) + b) \tag{2.30}$$

the VC-dimension is determined by the norm of the weight vector $\mathbf{w}$.

Let $\{(\mathbf{x_i}, y_i), \ldots, (\mathbf{x_l}, y_l)\}$ ($\mathbf{x_i} \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$) be the training set. Then, for a linearly separable training set we have:

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i = 1..l \quad . \tag{2.31}$$

The *margin* between the positive and negative points is defined by two hyperplanes $\mathbf{x} \cdot \mathbf{w} + b = \pm 1$ where the above term actually is zero. Fig. 2.5 illustrates this. Further, no points lie between these hyperplanes and the width of the margin is $2/||\mathbf{w}||$. The support vector algorithm now tries to maximize the margin by minimizing $||\mathbf{w}||$, which is supposed to be an optimal solution, *i.e.* where generalization is maximal. Once the maximum margin is obtained, data points lying on one of the separating hyperplanes, *i.e.* for which equation 2.31 yields zero, are called *support vectors* (illustrated by double circles in Fig. 2.5).

To simplify the calculation, the problem is formulated in a Lagrangian framework (see [246] for details). This leads to the maximization of the Lagrangians:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \, \mathbf{x_i} \cdot \mathbf{x_j} \tag{2.32}$$

subject to

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x_i} \quad , \tag{2.33}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad \text{and} \tag{2.34}$$

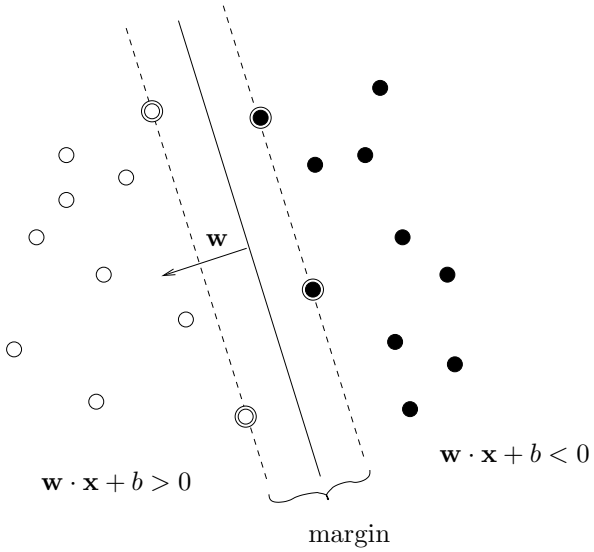


Figure 2.5: Graphical illustration of a linear SVM

$$\alpha_i \geq 0 \quad \forall i = 1..l \quad , \tag{2.35}$$

where $\alpha_i \, (i = 1..l)$ are the Lagrangian multipliers that are to be determined. Further, the solutions to α_i and condition 2.31 imply a value for b. Note that all α_i are zero except those corresponding to the support vectors.

Finally, new examples can simply be classified using the decision function 2.30.

In many cases, however, the training data cannot be completely separated because of some "outliers". Then, we might simply loosen the constraint 2.31 by introducing the constants $\xi_i > 0$ in the following way:

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) \geq (1 - \xi_i) \quad \forall i = 1..l \quad , \tag{2.36}$$

and condition 2.35 becomes

$$0 \leq \alpha_i \leq \xi_i \quad \forall i = 1..l \quad . \tag{2.37}$$

2.6.3 Non-linear Support Vector Machines

In order to use a *non-linear* decision function, the above formulas can quite easily be generalized. Boser *et al.* [23] proposed a simple method based on the so-called *kernel trick*. That is, before applying the dot product $x_i \cdot x_j$ in equation 2.32 the d-dimensional data is projected into a higher dimensional space where it is supposed to be linearly separable. Thus, a function $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$ is defined and $x_i \cdot x_j$ becomes $\Phi(x_i) \cdot \Phi(x_j)$. Now, instead of calculating Φ each time we use a kernel function $K(\mathbf{x_i}, \mathbf{x_j}) = \Phi(x_i) \cdot \Phi(x_j)$, *i.e.* each occurrence of the dot product is replaced by $K(\cdot, \cdot)$. Thus, if we want to classify a new data point $\mathbf{s}$ the decision function

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \, \mathbf{x_i} \cdot \mathbf{s} + b\right) \tag{2.38}$$

becomes

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \, \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{s}) + b\right) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \, K(\mathbf{x_i}, \mathbf{s}) + b\right). \tag{2.39}$$

With the kernel function K we don't need to calculate Φ or $\mathcal{H}$ but we must know if for a given K there *exists* a mapping Φ and some space $\mathcal{H}$ in which K is the dot product $K(\mathbf{x_i}, \mathbf{x_j}) = \Phi(x_i) \cdot \Phi(x_j)$. This property is ensured by the *Mercer's condition* [246]:

Theorem 1 *There exists a mapping Φ and an expansion*

$$K(\mathbf{x}, \mathbf{y}) = \sum_k \Phi(x)_k \Phi(y)_k \tag{2.40}$$

if and only if, for any $g(\mathbf{x})$ such that

$$\int g(\mathbf{x})^2 \, \mathrm{d}\mathbf{x} \quad \textit{is finite} \tag{2.41}$$

then

$$\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) \, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \geq 0 \quad . \tag{2.42}$$

Some examples for which the condition is satisfied are:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^n \quad \text{polynomial kernels} \tag{2.43}$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma ||\mathbf{x}-\mathbf{y}||^2} \quad \text{Gaussian radial basis function (RBF) kernels} \tag{2.44}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) - \delta) \quad \text{sigmoid kernels} \tag{2.45}$$

2.6.4 Extension to multiple classes

Up to this point, we only considered two-class problems. However, there are simple ways to extend the SVM method to several classes. One approach, called *one-against-all*, consists in training one classifier for each class that distinguishes between the examples of that class and the examples of all other classes. Thus, the number of SVMs equals the number of classes n.

Another approach trains a SVM for each possible pair of classes. To classify an example, it is input to each SVM and the class label corresponding to the maximal number of "winning" SVMs represents the final answer. The number of classifiers needed by this approach is $n(n-1)/2$, which is a drawback in terms of complexity compared to the first approach.

2.7 Bag of Local Signatures

As opposed to SVMs, being a very general classification technique, an approach called *Bag-of-Local-Signatures* (BOLS) has recently been introduced by Csurka *et al.* [51] for image classification problems, particularly object detection and

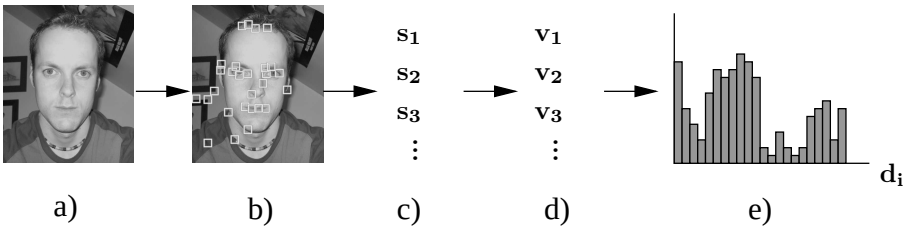


Figure 2.6: The histogram creation procedure with the Bag-of-local-signature approach: *a)* input image $\mathbf{I}$, *b)* detected salient points, *c)* extracted local signatures, *d)* quantized vectors (dictionary entries), *e)* histogram $h(\mathbf{I})$.

recognition. It was motivated by the *bag-of-words* approach for text categorization which simply counts the number of pre-defined key words in a document in order to classify it into one of several categories.

In the first step of the BOLS method, n salient points $\mathbf{p_i} = (x_i, y_i)$ of the input image are detected using an interest point detection algorithm, *e.g.* the Harris affine detector [162]. The small image region around each detected point is then represented by some local descriptors, such as the Scale-Invariant Feature Transform (SIFT) descriptors [148], leading to a local signature $\mathbf{s_i}$ for each salient point.

In the following step, the extracted signatures are classified applying any kind of vector quantization method. To this end, a dictionary of k representative signatures $\mathbf{d_j}$ ($j = 1..k$) is calculated from the training set using a clustering algorithm. For example, Csurka *et al.* [51] used the k-means clustering algorithm and Ros *et al.* [199] used a Self-Organizing Map (SOM).

Thus, for an image $\mathbf{I}$ to classify a *bag of local signatures* $\mathbf{v_i}$, *i.e.* entries of the dictionary, is obtained representing the appearance of the object in the image. However, for two different images of the same object the respective representations might differ due to the varying appearance in different views or partial occlusions making an efficient comparison difficult.

Therefore, discrete histograms $h(\mathbf{I})$ of the bag of local signatures $\mathbf{v_i}$ are calculated by simply counting the number of occurrences of the respective signatures. Finally, the histograms can be classified by using classical histogram distance measures, such as χ^2 or the Earth Mover's Distance (EMD) or by training a classifier on the vectors obtained from the histogram values, such as a Bayes classifier or SVMs [51].

Figure 2.6 illustrates the overall procedure for generating the Bag-of-Local-Signatures representation. A major advantage of this approach compared to statistical projection methods, for example, is its robustness to partial occlusions and to changing pose of the object to recognize. This is due to the purely *local* representation and the rotation- and scale-invariant description of the local image patches.

As this technique is a relatively new approach in the field of machine learning and very specific to image classification we won't describe it here in more detail. We will rather concentrate on a very versatile and powerful machine learning technique constituting the basis for all of the face analysis approaches proposed in this work, namely *Artificial Neural Networks*.

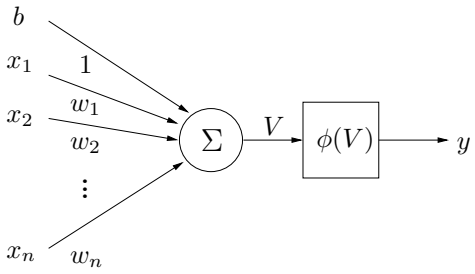


Figure 2.7: The Perceptron

2.8 Neural Networks

2.8.1 Introduction

Artificial Neural Networks (ANN), short Neural Networks (NN), denote a machine learning technique that has been inspired by the human brain and its capacity to perform complex tasks by means of inter-connected neurons performing each a very simple operation. Likewise, a NN is a trainable structure consisting of a set of inter-connected units, each implementing a very simple function, and together eventually performing a complex classification function or approximation task.

2.8.2 Perceptron

The most well known type of neural unit is called *Perceptron* and has been introduced by Rosenblatt [200]. Its basic structure is illustrated in Fig. 2.7. It has n inputs and one output where the output is a simple function of the sum of the input signals $\mathbf{x}$ weighted by $\mathbf{w}$ and an additional bias b. Thus,

$$y = \phi(\mathbf{x} \cdot \mathbf{w} + b) \quad . \tag{2.46}$$

Often, the bias is put inside the weight vector $\mathbf{w}$ such that $w_0 = b$ and the input vector $\mathbf{x}$ is extended correspondingly to have $x_0 = 1$. Equation 2.46 then becomes:

$$y = \phi(\mathbf{x} \cdot \mathbf{w}) \quad . \tag{2.47}$$

where ϕ is the Heavyside step function:

$$\begin{aligned} \phi &: \mathbb{R} \to \mathbb{R} \\ \phi(x) &= \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{2.48}$$

The Perceptron thus implements a very simple two-class classifier where $\mathbf{w}$ is the separating hyperplane such that $\mathbf{w} \cdot \mathbf{x} \geq 0$ for examples from one class and $\mathbf{w} \cdot \mathbf{x} < 0$ for examples from the other.

In 1962, Rosenblatt introduced the *perceptron convergence theorem* [201], a supervised training algorithm capable of learning arbitrary two-class classification problems. However, Minsky and Papert [163] pointed out that there are very simple classification problems where the perceptron fails, namely when the two classes are not linearly separable like in the XOR-problem, where the

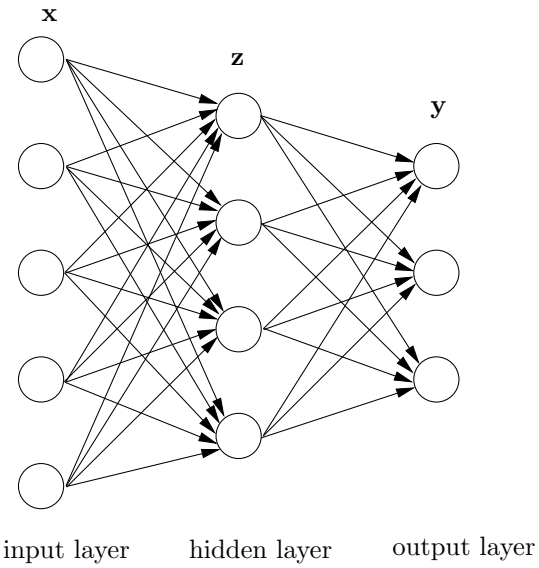


Figure 2.8: A Multi-Layer Perceptron

pattern $(0,0)$ and $(1,1)$ belong to one class and $(0,1)$ and $(1,0)$ to the other. This motivated the use of several interconnected perceptrons which are able to form more complex decision boundaries by combining several hyperplanes. The most common type of these NNs is the Multi-Layer Perceptron described in the following section.

2.8.3 Multi-Layer Perceptron

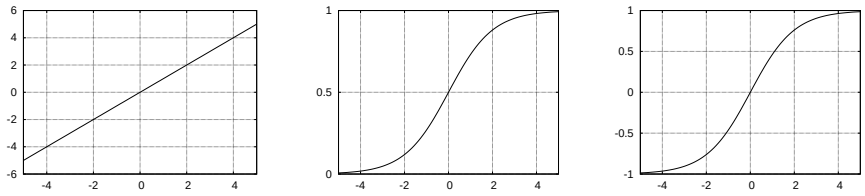
Multi-Layer Perceptrons (MLP) are capable of approximating arbitrarily complex decision functions. With the advent of a practicable training algorithm in the 1980's, the so-called *Backpropagation algorithm* [208], they became the most widely used form of NNs.

Fig. 2.8 illustrates the structure of a MLP. There is an input layer, one or more hidden layer(s) and an output layer of neurons, where each neuron except the input neurons implements a perceptron as described in the previous section. Moreover, the neurons of one layer are only connected to the following layer. We call this type of network: *feed-forward* network, *i.e.* the activation of the neurons is propagated layer-wise from the input to the output layer. And if there is a connection from each neuron to every neuron in the following layer, as in Fig. 2.8, the network is called *fully connected*. Further, the neurons' activation function has to be differentiable in order to adjust the weights by the Backpropagation algorithm. Commonly used activation functions are for example:

$$\phi(x) = x \qquad \text{linear} \tag{2.49}$$

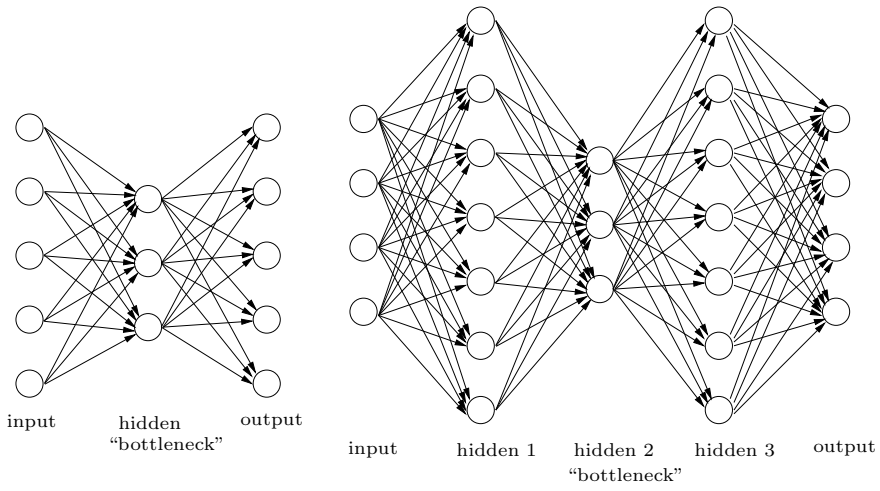
$$\phi(x) = \frac{1}{1+e^{-cx}} \quad (c>0) \qquad \text{sigmoid} \tag{2.50}$$

$$\phi(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad \text{hyperbolic tangent.} \tag{2.51}$$



(a) linear (b) sigmoid (c) hyperbolic tangent

Figure 2.9: Different types of activation functions



(a) 3-layer AANN (b) 5-layer AANN

Figure 2.10: Auto-Associative Neural Networks

Figure 2.9 shows the three types of functions. Note that the linear function is in the range $]-\infty, +\infty[$, the sigmoid function in $]0, +1[$ and the hyperbolic tangent function in $]-1, +1[$. The linear activation function is mostly bounded by a maximum and minimum value, *e.g.* -1 and $+1$, and thus it becomes a step-wise linear function. However, when using the Backpropagation learning algorithm (explained in section 2.8.5) one has to be careful with the points where the step-wise function is not differentiable, *e.g.* $x = -1$ and $x = +1$.

2.8.4 Auto-Associative Neural Networks

Auto-Associative Neural Networks (AANN) [47, 46, 86] are a special type of MLP where the hidden layer has fewer neurons than the input and output layers, a so-called "bottle-neck". The NN is trained to reproduce the input pattern at its output. Thus, input and output layer have the same dimension. Figure 2.10(a) illustrates this architecture. When training the AANN to reconstruct the input patterns, it will actually learn a mapping into a lower-dimensional space and the respective inverse mapping. Thus, the hidden layer learns a compact representation of the data, and this technique can be applied to data

compression. If the neurons' activation functions are linear, it has been shown by Baldi and Hornik [9] that the projection performed by this type of NN is equivalent to a PCA of the input data's covariance matrix, and the weight vectors of the output neurons correspond to its leading eigenvectors. However, if the activation functions are *non-linear* [47] this is *not* true as proven by Japkowitz *et al.* [114]. In that case, a non-linear projection is learned which is still closely connected to the theoretical PCA [24]. Further, as pointed out by Bourlard *et al.* [24] and Funahashi [79], the compression with non-linear activation functions is not superior to the linear one.

A superior architecture is composed of three hidden layers as illustrated in Fig. 2.10(b) and has been used many times in the literature [45, 244, 174, 125, 168]. The activation functions of the first and third hidden layer are non-linear. These layers are referred to as *mapping layer*. The second hidden layer has either a linear or non-linear activation function and is called *representation layer*.

As the 3-layer AANN with sigmoid activation function, this NN performs a non-linear dimensionality reduction. By means of the Backpropagation algorithm, it can thus learn any non-linear mapping from the input space to a lower-dimensional sub-space and the respective reconstruction. However, a major drawback is the manual choice of the subspace's *dimension*, *i.e.* the number of hidden neurons in the "bottleneck" layer, before training the NN.

2.8.5 Training Neural Networks

In general, the parameters of a NN, *i.e.* the weights and biases, are learned using a training data set. However, as the space of possible weights can be very large and of high dimension the analytical determination of these weights might be very difficult or even infeasible. For this reason, an *iterative* approach is adopted in most cases.

There are two principal training modes which determine the way the weights are updated:

Online training: After presentation of each training example, the error is calculated, and the weights are updated accordingly.

Offline training: The whole training set is propagated through the NN, and the respective errors are accumulated. Finally, the weights are updated using the accumulated error. This is also called *batch* training.

Many different NN training algorithms have been published in the literature. Some work only in online, some only in offline mode, and some can be executed in both ways. Which algorithm is best for a given problem depends on the NN architecture, the nature and cardinality of the training data set and the type of function to learn. Therefore, there is no basic rule for the choice of the training algorithm.

In the following, we will focus on the *Backpropagation algorithm* since it is the most common and maybe most universal training algorithm for feed-forward NNs, especially for MLPs. Some alternative methods will also be presented at the end of this section.

The Backpropagation algorithm

In the context of NNs, the Backpropagation (BP) algorithm has initially been presented by Rumelhart *et al.* [208]. It is a supervised learning algorithm defining an error function E and applying the gradient descent technique in the weight space in order to minimize E. The combination of weights leading to a minimum of E is considered to be a solution of the learning problem. Note that the BP algorithm does not guarantee to find a global minimum which is an inherent problem of gradient descent optimization. However, we will discuss some approaches to overcome this problem in the following section. In order to calculate the gradient of E, at each iteration, the error function has to be continuous and differentiable. Thus, the activation function of each individual perceptron must also have this property as mentioned in section 2.8.3. Mostly, a sigmoid or hyperbolic tangent activation function is employed, depending on the range of desired output values, *i.e.* $]0, 1[$ or $]-1, +1[$. Note that BP can be performed in *online* or *offline* mode, *i.e.* E represents either the error of one training example or the sum of errors produced by all training examples.

In the following, we will explain the standard online BP algorithm, also known as Stochastic Gradient Descent, applied to MLPs. There are two phases of the algorithm:

- the *forward* pass, where a training example is presented to the network and the activations of the respective neurons is propagated layer by layer until the output neurons.
- the *backward* pass, where at each neuron the respective error is calculated starting from the output neurons and, layer by layer, propagating the error back until the input neurons.

Now, let us define the error function as:

$$E = \frac{1}{2} \sum_{p=1}^{P} ||\mathbf{o_p} - \mathbf{t_p}||^2 \quad , \tag{2.52}$$

where P is the number of training examples, $\mathbf{o_p}$ are the output values produced by the NN having presented example p, and $\mathbf{t_p}$ are the respective target values. The goal is to minimize E by adjusting the weights of the NN. With *online* learning we calculate the error and try to minimize it after presenting each training example. Thus,

$$E_p = \frac{1}{2}||\mathbf{o_p} - \mathbf{t_p}||^2 = \frac{1}{2} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2 \quad , \tag{2.53}$$

where K is the number of output units. When minimizing this function by gradient descent, we calculate the steepest descent of the error surface in the weight space, *i.e.* the opposite direction of the gradient $\nabla E_p = \left(\frac{\partial E_p}{\partial w_1}, \ldots, \frac{\partial E_p}{\partial w_k}\right)$. In order to ensure convergence, the weights are only updated by a proportion of the gradient. Thus,

$$\Delta w_k = -\lambda \frac{\partial E_p}{\partial w_k} \tag{2.54}$$

Now, let us define

$e_{pk} = o_{pk} - t_{pk}$	the error of pattern p at output neuron k
w^o_{kj}	the weight from hidden neuron j to output neuron k
w^h_{ji}	the weight from input neuron i to hidden neuron j
$z_{pj} = \phi\left(\sum_i w_{ji}\, x_{pi}\right)$	the output of hidden neuron j
$V_{pk} = \sum_j w^o_{kj}\, z_{pj}$	the weighted sum of all inputs z_{pj} of output neuron k
$V_{pj} = \sum_i w^h_{ji}\, x_{pi}$	the weighted sum of all inputs x_{pi} of hidden neuron j.

Note that for simplicity we consider only one hidden layer containing J neurons (as in Fig. 2.8). The subscript p always refers to pattern p, i refers to input neuron i, j to hidden neuron j and k to output neuron k. By applying the chain rule to equation 2.54, for one particular weight w^o_{kj} and training example p, we have:

$$\begin{aligned}
\Delta w^o_{kj} &= -\lambda\, \frac{\partial E_p}{\partial w^o_{kj}} \\
&= -\lambda\, \frac{\partial E_p}{\partial e_{pk}}\, \frac{\partial e_{pk}}{\partial o_{pk}}\, \frac{\partial o_{pk}}{\partial V_{pk}}\, \frac{\partial V_{pk}}{\partial w^o_{jk}} \\
&= -\lambda\, e_{pk}\, \phi'(V_{pk})\, z_{pj} \\
&= -\lambda\, \delta_{pk}\, z_{pj} \quad ,
\end{aligned}$$

where

$$\delta_{pk} = e_{pk}\, \phi'(V_{pk}) \tag{2.55}$$

is the local gradient. This holds for output neurons. For the hidden neurons the respective equations are slightly different:

$$\begin{aligned}
\Delta w^h_{ji} &= -\lambda\, \frac{\partial E_p}{\partial w^h_{ji}} \\
&= -\lambda\, \frac{\partial E_p}{\partial z_{pj}}\, \frac{\partial z_{pj}}{\partial V_{pj}}\, \frac{\partial V_{pj}}{\partial w^h_{ji}} \\
&= -\lambda\, \left(\sum_{k=1}^{K} e_{pk}\, \frac{\partial e_{pk}}{\partial z_{pj}}\right)\, \phi'(V_{pj})\, x_{pi} \\
&= -\lambda\, \left(\sum_{k=1}^{K} e_{pk}\, \frac{\partial e_{pk}}{\partial o_{pk}}\, \frac{\partial o_{pk}}{\partial V_{pk}}\, \frac{\partial V_{pk}}{\partial z_{pj}}\right)\, \phi'(V_{pj})\, x_{pi} \\
&= -\lambda\, \left(\sum_{k=1}^{K} e_{pk}\, \phi'(V_{pk})\, w^o_{kj}\right)\, \phi'(V_{pj})\, x_{pi} \\
&= -\lambda\, \delta_{pj}\, x_{pi}
\end{aligned}$$

where

$$\delta_{pj} = \left(\sum_{k=1}^{K} \delta_{pk} w^o_{kj}\right)\, \phi'(V_{pj}) \tag{2.56}$$

is the local gradient for hidden neuron j $(j = 1..J)$. Algorithm 3 summarizes the standard online Backpropagation algorithm. The respective variables are noted as functions of iteration n, *e.g.* $w_{kj}(n)$. ϵ is a small constant that determines the convergence criteria and max_{iter} is the maximum number of iterations.

Algorithm 3 The standard online Backpropagation algorithm for MLPs

Initialize all weights of the NN to some small random value
Set the learning rate λ to a small positive value
$n = 1$
repeat
  for $p = 1$ to P **do**
    propagate pattern $\mathbf{x_p}$ through the network
    compute the weight changes for the K output neurons:
      $\Delta w^o_{kj}(n) = -\lambda\, \delta_{pk}(n)\, z_{pj}(n)$ with $\delta_{pk}(n) = e_{pk}(n)\, \phi'(V_{pk}(n))$
    compute the weight changes for the J hidden neurons:
      $\Delta w^h_{ji}(n) = -\lambda\, \delta_{pj}(n)\, x_{pi}(n)$
        with $\delta_{pj}(n) = \left(\sum_{k=1}^{K} \delta_{pk}(n)\, w^o_{kj}(n)\right) \phi'(V_{pj}(n))$
    update all weights: $\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n)$
    $n = n + 1$
  end for
  $E = \frac{1}{2} \sum_{p=1}^{P} ||\mathbf{o_p} - \mathbf{t_p}||^2$
until $E < \epsilon$ or $n > max_{\text{iter}}$

The extension of the described algorithm to *several* hidden layers is straightforward and will not be discussed further. In the next section, we will rather concentrate on some common improvements to the Backpropagation algorithm.

Extensions to Backpropagation

Many enhancements to standard Backpropagation have been proposed in the literature. They aim mainly at improving the convergence of the algorithm, avoiding local minima or improving the generalization of the NN.

Momentum In order to take into account previous weight changes, a *momentum term* [186] can be added to the weight update formula, *i.e.*

$$\Delta w_k(n) = -\lambda \frac{\partial E_p}{\partial w_k(n)} + \underbrace{\alpha \Delta w_k(n-1)}_{\text{momentum term}} \quad , \tag{2.57}$$

where $0 \leq \alpha < 1$ is the momentum rate. The objective of the momentum term is to improve the convergence speed. If there are several consecutive weight updates in the same direction the weight changes will be bigger, and, on the contrary, if the weight vector oscillates the changes will be smaller or even zero. Thus, the momentum term causes some kind of smoothing or averaging over time (*i.e.* over the iterations) regarding the weight updates.

Weight decay This is a regularization technique introduced by Werbos *et al.* [257] where the term $\frac{\alpha}{2} \sum_x w_x^2$ is added to the error function. The w_x are all

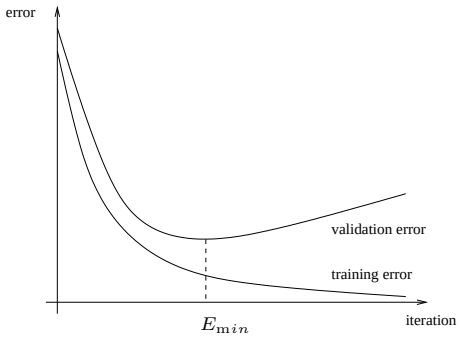


Figure 2.11: Typical evolution of training and validation error

the weights and biases of the NN, and α is a small positive constant to choose beforehand. Thus,

$$E_p = \frac{1}{2}||\mathbf{o_p} - \mathbf{t_p}||^2 + \frac{\alpha}{2}\sum_x w_x^2 \quad . \tag{2.58}$$

This term penalizes large weights and reduces the flexibility of the NN. The goal here is to improve the generalization capacity of the NN and to avoid overfitting to the data. Performing gradient descent on this error function leads to the following weight update formula:

$$\Delta w_x = -\lambda \frac{\partial E_p}{\partial w_x} - \lambda\,\alpha\,w_x \quad . \tag{2.59}$$

This means, at each iteration the weights are decreases by a proportion of their value unless they are reinforced by Backpropagation. The effect is similar to pruning where useless weights are set to zero.

Cross-validation A common technique used to improve the generalization capacity of a NN is called *cross-validation*. Here, the training set is divided into two disjoint parts. One used for the actual training and the other for validation, *i.e.* to verify how well the NN performs on unknown data. In most cases it can be noticed that both the error on the training and validation set decrease at the beginning of training but at some point the validation error stays constant or even increases (as shown in Fig. 2.11). It is assumed that at this point the NN starts to overtrain the data and generalization decreases. Thus, the training is stopped when the validation error is minimal. This technique is commonly known as "early stopping".

Determining the learning rate

The choice of the learning rate λ used when updating the weights is crucial for the successful convergence and the generalization capacity of the network. A too small learning rate leads to slow convergence and a too high learning rate to divergence. Moreover, in the latter case the network is likely to overfit to the training data when using an *online* Backpropagation algorithm as it might specialize to the examples presented at the beginning of the training. Numerous

solutions for the dynamic adaptation of the learning rate have been proposed in the literature. Most of them focus on the acceleration of the training process rather than their generalization performance. They can roughly be divided into two groups: global and local adaption techniques. The former is referring to methods adjusting an overall learning rate for the whole network and the latter to the adaptation of independent learning rates for each weight.

A method for global adaptation has been proposed by Chan *et al.* [37] where the angle between the last weight update and the current gradient is calculated. If it is less than 90° the learning rate is increased otherwise it is decreased. Salomon *et al.* [213] proposed an evolutionary based adaption of the learning rate. At each iteration, two weight updates, one with increased and the other with decreased learning rate, are performed separately. The resulting network that performs better is retained and used as a starting point for the next iteration. A heuristic method, the so-called "bold driver" method, has been employed by Battiti *et al.* [14] and Vogl *et al.* [255]. Here the learning rate is adjusted according to the evolution of the error criteria E. If E decreases the learning rate is slightly increased, otherwise it is drastically decreased. Hsin *et al.* [102] proposed to use a weighted average of the cosines between successive weight updates, and Plagianakos *et al.* [179] calculated a two point approximation to the secant equation underlying quasi-Newton methods in order to obtain a dynamic learning rate and additionally make use of an acceptability condition to ensure convergence. Finally, the approach of Magoulas *et al.* [151] estimates the local shape of the error surface by the Lipschitz constant and sets the learning rate accordingly. They also employed this technique to calculate a separate dynamic learning rate for each weight [152].

Local learning rate adjustment methods have been very popular due to their efficiency and generally higher convergence speed. A very well-know technique is the Delta-Bar-Delta method introduced by Jacobs *et al.* [112]. Here, the learning rates are adjusted according to sign changes of the exponential averaged gradient. Similarly, Silva and Almeida [226] proposed a method where the learning rates are increased if the respective gradients of the last two iterations have the same sign and decreased otherwise.

Finally, many methods do not use an explicit learning rate but first calculate a descent gradient direction and then perform a line search (*c.f.* Alg. 6) such that the error criteria is minimized in the direction of the gradient [149, 63, 140].

Note that most of the existing adaptive learning algorithms are *offline* learning algorithms However, *online* algorithms generally converge faster when the input space is large compared to the number of examples (e.g. in image processing tasks) or in more complex architectures like Convolutional Neural Networks (CNN) that use shared weights. Thus, for many real world applications the online Backpropagation algorithm or its variants are still the best choice. There are also some adaptive online algorithms in the literature. For example, Schraudolph [220], Harmon *et al.* [95] and Almeida *et al.* [1] proposed methods similar to the Incremental Delta-Bar-Delta approach introduced by Sutton *et al.* [232], an extension of the Delta-Bar-Delta technique for online training.

In the following section we will propose a novel approach to learning rate adaption with online Backpropagation. We will show that our algorithm not only increases convergence speed but also improves the generalization capacity of the resulting NN.

A novel approach to learning rate adaption with online Backpropagation

We now introduce a new method to automatically adapt the learning rate of the online Backpropagation algorithm [57]. The proposed algorithm requires a separation of the data set into training and validation set, as described on page 31. But here the validation set is not only used for "early stopping" but also to control the learning rate, *i.e.* to accelerate or decelerate learning. The learning rate is now denoted as a function of n, denoting the iteration, *i.e.* $\lambda(n)$. Thus, it is adjusted only after each iteration and not after each training example.

The algorithm consists of two consecutive phases:

1. the main learning phase and
2. the refinement phase.

The main learning phase The adaption of the learning rate in our approach is similar to the "bold driver" method [255, 14] where the learning rate is initialized with a very small value (e.g. 10^{-10}) and adjusted after each training iteration according the difference of the error criteria E between the current and the preceding iteration.

The proposed method applies this idea to the validation set instead of the training set in order to reduce overfitting. Moreover, the procedure is slightly modified to be more tolerant to error increases as the validation error is more likely to oscillate than the training error. Let us consider the typical shapes of the error curves of a training and validation set when using standard Backpropagation. Fig. 2.11 illustrates this in a simplified manner. When applying the technique of "early stopping" the weight configuration at iteration $n_{\min}$, i.e. where the validation error is minimal, is retained as the network is supposed to show the highest generalization performance at this point. Further training likely leads to overfitting. The purpose of the first phase of the proposed algorithm is thus to reach the point $n_{\min}$ more quickly.

To this end, the normalized difference between the error criteria of the current and the preceding iteration is calculated:

$$\delta(n) = \frac{E_v(n) - E_v(n-1)}{E_v(n)} \quad . \tag{2.60}$$

$E_v(n)$ is the error criteria at iteration n calculated on the whole validation set (*c.f.* Eq. 2.52).

The algorithm further requires a running average $\bar{\delta}(n)$ of the preceding values of δ:

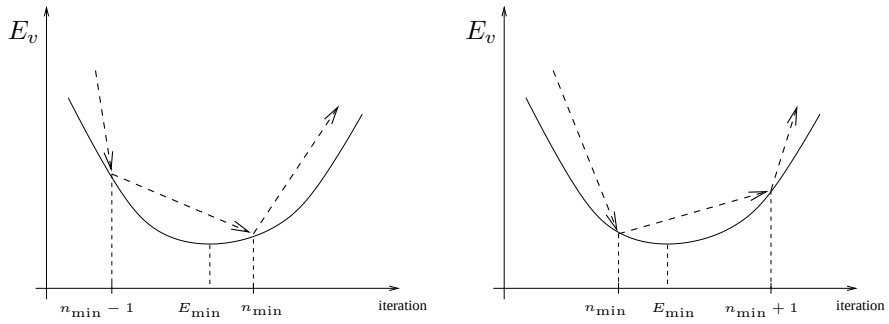
$$\bar{\delta}(n) = \alpha \cdot \delta(n) \; + \; (1-\alpha) \cdot \bar{\delta}(n-1) \quad , \tag{2.61}$$

where $0 < \alpha \leq 1$ (e.g. $\alpha = 0.1$).

The principal learning rate updating rule is the following:

$$\lambda(n) = \begin{cases} d \cdot \lambda(n-1) & \text{if } \; \delta(n) \cdot \bar{\delta}(n-1) < 0 \; \text{ and } \; |\bar{\delta}(n-1)| > \theta \; , \\ u \cdot \lambda(n-1) & \text{otherwise} \; . \end{cases} \tag{2.62}$$

where u and d are positive constants, $0 < d < 1 < u$, and θ is a threshold to allow for small error oscillations. In our experiments we used $u = 1.1$, $d = 0.5$



(a) The actual minimum has been missed *before* iteration $n_{\min}$

(b) The actual minimum has been missed *after* iteration $n_{\min}$

Figure 2.12: The two possible cases that can occur when the minimum on the validation set is reached

and $\theta = 0.01$. Thus, the learning rate adaption is based on the signs of the error differences of the current and the preceding iterations. If the sign changes the learning rate is decreased, otherwise it is increased. The principle of this procedure is similar to the Delta-Bar-Delta method [112] but the calculation is not based on gradients.

Refinement phase If the training has passed iteration $n_{\min}$ where E_v is minimal the network is likely to overtrain. In fact, as the gradient descent is performed in discrete steps the actual minimum $E_{\min}$ of the error surface of the validation set is likely to be missed and lies between the weight configurations of two successive training iterations. Fig. 2.12 illustrates this. Clearly, there are two cases to differentiate: the minimum $E_{\min}$ has been missed either before or after iteration $n_{\min}$. Now we assume that the validation error surface is relatively smooth and that no other local minimum lies between iterations $n_{\min}-1$ and $n_{\min}$ or between iterations $n_{\min}$ and $n_{\min}+1$ respectively. In order to try to attain a smaller error the network reverts to the weight configuration at iteration $n_{\min} - 1$, decreases the learning rate, and training is continued. Note that for training only the examples of the training set are used. Thus, it is uncertain if the actual minimum can be attained at all. If no smaller error has been found for a certain number of iterations N the "real" minimum, *i.e.* the minimum on E_v, is more likely to have occurred "after" iteration $n_{\min}$, (see Fig. 2.12(b)). In this case, the network reverts to iteration $n_{\min}$, the learning rate is again decreased and training continues. If a smaller error is reached during this process, the temporary minimum is retained, and the training continues normally. Otherwise, the reverting procedure is repeated while always retaining the absolute minimum and the respective weight configuration found so far. Algorithm 4 summarizes the overall training procedure. Note that the computational overhead of the algorithm compared to standard Backpropagation with fixed learning rate is negligible as the error on the validation set needs to be calculated anyway if cross-validation is performed.

Fig.2.13 illustrates a typical evolution of the error criteria $E_v(n)$ during the training process using the proposed learning algorithm. Due to the initialization with a very small value, the error stays nearly constant at the beginning but

Algorithm 4 The proposed online Backpropagation algorithm with adaptive learning rate

Initialize weights and individual learning rates
Set $\lambda(0) := 10^{-10}$, $\delta(0) := 0$ and $\bar{\delta}(0) := 0$
Calculate $E_v(0)$
$E_{\min} = E_v(0)$
$n := 0$
repeat
  Do one training iteration
  $n := n + 1$
  Calculate $\delta(n) = \frac{E_v(n) - E_v(n-1)}{E_v(n)}$
  if $\delta(n) \cdot \bar{\delta}(n-1) < 0$ and $|\bar{\delta}(n-1)| > \theta$ **then**
    $\lambda(n) = d \cdot \lambda(n-1)$
  else
    $\lambda(n) = u \cdot \lambda(n-1)$
  end if
  $\bar{\delta}(n) = \alpha \cdot \delta(n) + (1 - \alpha) \cdot \bar{\delta}(n-1)$
  if $E_v(n) < E_{\min}$ **then**
    save the current weight configuration
    $E_{\min} = E_v(n)$
    $n_{\min} := n$
  end if
  if $n - n_{\min} > N$ **then**
    Revert to weight configuration at $n_{\min} - 1$ (or $n_{\min}$)
  end if
until $n = n_{\max}$

drops very quickly at some point due to the exponential increase of the learning rate, and finally it converges to a minimum. In general, the main part of the minimization is done in the first phase, and the error decrease in the refinement phase is relatively small.

In order to validate the approach and to show the benefit compared to standard BP and the basic "bold driver" method we will present some experimental results hereafter.

Experimental Results We evaluated the proposed learning algorithm on a MLP trained to classify the examples of the well-known NIST database of handwritten digits. The database contains 3823 training and 1797 test examples of 8x8 matrices. From the training set 500 examples were selected randomly and used for validation. The MLP we used for the experiments had 64 input, 10 hidden and 10 output neurons, fully inter-connected. The neurons all had sigmoid activation functions.

To ensure that the neural network is well-conditioned, we additionally used *fixed local* learning rates that were distributed stepwise from the last layer to

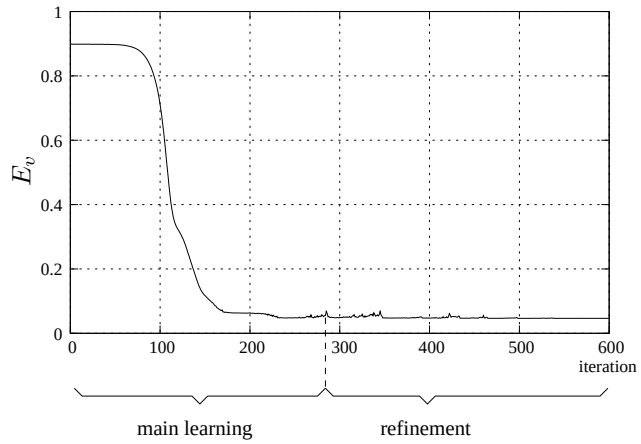


Figure 2.13: A typical evolution of the error criteria on the validation set using the proposed learning algorithm

the first layer according to the incoming connections of each neuron. Thus, the output neurons had the highest and the input the lowest local learning rate. The overall learning rate is just the product of the fixed local and the dynamic global learning rate.

In the first experiment, we compared the convergence properties of the proposed algorithm to the ones of standard Backpropagation. Fig. 2.14 shows the resulting error curves evaluated on the validation set. The different curves for the Backpropagation algorithm have been obtained by using different global learning rates ($10^{-3}, 10^{-4}$ and 10^{-5}). The global learning rate of the proposed method was initialized with the value 10^{-7}. Note that our approach converges more slowly at the beginning but catches up quickly and finishes stable on the same level or even lower than Backpropagation.

Fig. 2.15 illustrates that our method is not sensitive to different initializations of the global learning rate. The curves show the validation error curves for three different runs with initial learning rates of 10^{-6}, 10^{-8} and 10^{-10} respectively. Note that the point of time where the minimum is reached increases only *linearly* when the initial learning rate is decreased *exponentially*. This is another side effect of the exponential learning rate update rule. All the runs converge to approximately the same solution, and the recognition rates are about the same for all networks.

The final experiment demonstrates that the algorithm not only converges faster but also improves the generalization performance of the resulting neural networks. To this end, the training set was gradually reduced and the respective recognition rates on the test set were calculated and compared to the standard Backpropagation as well as to the bold driver method [255, 14]. Table 2.1 shows the overall results. One can see that the proposed method performs slightly better with training set sizes 3323 and 1000 and clearly outperforms the other algorithms when only 600 and 100 training examples are used.

Table 2.2 shows the respective results with a neural network with 40 hidden neurons. The recognition rates of the proposed method are slightly better then for the other algorithms, although the difference is less significant than with

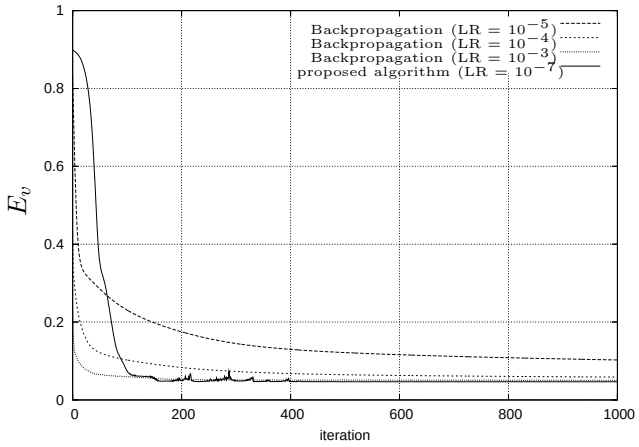


Figure 2.14: The evolution of the validation error on the NIST database using Backpropagation and the proposed algorithm

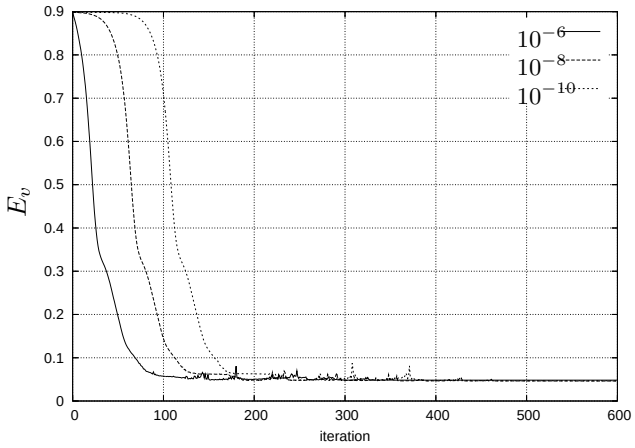


Figure 2.15: The validation error curves of the proposed approach with different initial global learning rates

training set size / algorithm	3323	1000	600	100
Backpropagation	94.30	93.42	78.31	73.88
bold driver [255, 14]	93.41	91.32	83.74	72.75
proposed algorithm	94.50	93.71	**85.29**	**78.10**

Table 2.1: Recognition rate (in %) with varying training set size (10 hidden neurons)

training set size / algorithm	3323	1000	600	100
Backpropagation	95.71	93.89	86.58	80.31
bold driver [255, 14]	94.97	93.20	86.45	79.96
proposed algorithm	95.77	93.81	87.06	80.47

Table 2.2: Recognition rate (in %) with varying training set size (40 hidden neurons)

the smaller NN. However, convergence speed is still superior as illustrated in Fig. 2.14.

Other learning algorithms

Backpropagation is by far the most widely used training algorithm for feed-forward NNs but there are other possible solutions for searching for a weight configuration that minimizes the overall error of the NN. In the following, some of them will be briefly explained.

RPROP The *Resilient Backpropagation* (RPROP) learning algorithm introduced by Riedmiller *et al.* [197] uses a step size which doesn't depend on the gradient magnitude but which is increased or decreased according to gradient sign changes. RPROP is also an offline learning algorithm. Further, it uses an independent step size $\Delta_i(n)$ for each weight $w_i(n)$. Algorithm 5 describes the principal procedure of RPROP. Recommended values for the constants are: $\Delta_{\max} = 50,\ \Delta_{\max} = 50,\ u = 1.2,\ d = 0.5$.

Conjugate gradient algorithm This is a training algorithm based on line search and works only in offline mode. Algorithm 6 outlines the principal steps of a line search algorithm.

In the conjugate gradient algorithm, we have for the first search direction $\mathbf{d}(1) = -\nabla E(n)$, and each following direction $\mathbf{d}(n)$ $(\forall n > 1)$ satisfies:

$$\mathbf{d}(n)\,\mathbf{H}\,\mathbf{d}(n-1) = 0 \quad , \tag{2.71}$$

where $\mathbf{H}$ is the Hessian matrix whose components are defined as

$$H_{ij} = \frac{\partial^2 E}{\partial w_i\,\partial w_j} \quad . \tag{2.72}$$

Note that if $\mathbf{H}$ is the identity matrix two consecutive directions are orthogonal. $\mathbf{d}(n)$ is called the conjugate direction. It has the property that along this direction the gradient does not change its direction but only its length. Thus, moving along $\mathbf{d}$ does not affect the result of the previous iteration.

Calculating $\mathbf{H}$ is rather inefficient in more complex settings. For this reason, there are some practical methods to calculate $\mathbf{d}(n)$ without computing $\mathbf{H}$ explicitely but only by using $\mathbf{d}(n-1)$ and gradient information. The new search direction $\mathbf{d}(n)$ is given as:

$$\mathbf{d}(n) = -\nabla E(n) + \beta(n)\,\mathbf{d}(n-1) \quad , \tag{2.73}$$

Algorithm 5 The RPROP algorithm

$n = 1$
$\Delta_i(1) = \Delta_{\min} \quad \forall i = 1..N$
repeat
  Perform forward propagation of every example while accumulating $\nabla E(n)$
  for $i = 0$ to N **do**
    Update the step sizes:

$$\Delta_i(n) = \Delta_i(n-1) \cdot u \qquad \text{if } \frac{\partial E(n)}{\partial w_i(n)} \cdot \frac{\partial E(n-1)}{\partial w_i(n-1)} > 0 \tag{2.63}$$

$$\Delta_i(n) = \Delta_i(n-1) \cdot d \qquad \text{if } \frac{\partial E(n)}{\partial w_i(n)} \cdot \frac{\partial E(n-1)}{\partial w_i(n-1)} < 0 \tag{2.64}$$

$$\Delta_i(n) = \Delta_{\max} \qquad \text{if } \Delta_i(n) \geq \Delta_{\max} \tag{2.65}$$

$$\Delta_i(n) = \Delta_{\min} \qquad \text{if } \Delta_i(n) \leq \Delta_{\min} \tag{2.66}$$

    Update the weights:
    if $\frac{\partial E(n)}{\partial w_i(n)} \cdot \frac{\partial E(n-1)}{\partial w_i(n-1)} \geq 0$ **then**

$$\Delta w_i(n) = -\Delta_i(n) \qquad \text{if } \frac{\partial E(n)}{\partial w_i(n)} > 0 \tag{2.67}$$

$$\Delta w_i(n) = +\Delta_i(n) \qquad \text{if } \frac{\partial E(n)}{\partial w_i(n)} < 0 \tag{2.68}$$

    else

$$\Delta w_i(n) = 0 \tag{2.69}$$

    end if

$$w_i(n) = w_i(n-1) + \Delta w_i(n) \tag{2.70}$$

  end for
until $E < \epsilon$

Algorithm 6 The line search algorithm

n=1
repeat
  Determine a search direction $\mathbf{d}(n)$
  Minimize $E(n)$ w.r.t. a factor λ by modifying the weight vector $\mathbf{w}(n)$ along the search direction, *i.e.* $\tilde{\mathbf{w}}(n) = \mathbf{w}(n) + \lambda \mathbf{d}$. Let λ_{opt} be this value.
  Update the weight vector: $\mathbf{w}(n+1) = \mathbf{w}(n) + \lambda_{\text{opt}}\, \mathbf{d}$
  $n = n + 1$
until $E < \epsilon$ (ϵ being a small positive constant)

where $\beta(n)$ can be calculated by using, for example, the Fletcher-Reeves formula:

$$\beta(n) = \frac{\nabla E(n)^T \nabla E(n)}{\nabla E(n-1)^T \nabla E(n-1)} \tag{2.74}$$

or the Polak-Ribiere formula:

$$\beta(n) = \frac{(\nabla E(n) - \nabla E(n-1))^T \nabla E(n)}{\nabla E(n-1)^T \nabla E(n-1)} \quad . \tag{2.75}$$

It has been shown that in case of a quadratic polynomial error function the algorithm converges in N steps, where N is the number of weight parameters. However, in most applications the error function is not quadratic, hence the convergence is slower.

Newton algorithm In Newton's method the error surface E is assumed to be quadratic polynomial. Then, the minimum of E can be reached in one step (the so-called Newton step):

$$\Delta \mathbf{w} = -\mathbf{H}^{-1} \nabla E \quad , \tag{2.76}$$

where $\mathbf{H}$ is the Hessian matrix (*c.f.* Eq. 2.72). However, as E is usually non-quadratic only a small step in this direction is performed, and $\mathbf{H}$ is computed with the new weight configuration. Thus, at each iteration we calculate,

$$\Delta \mathbf{w} = -\lambda \mathbf{H}^{-1} \nabla E \quad , \tag{2.77}$$

where $0 < \lambda < 1$.

In most applications, the basic Newton's algorithm is impractical because of the following reasons:

- $\mathbf{H}$ has to be positive definite in order to allow calculating its inverse. However, this is not always the case.
- A $N \times N$ matrix must be stored and inverted at each iteration which is of $O(N^3)$ computational complexity.
- The quadratic local approximation of E is often two imprecise such that λ has to be chosen very small.

A partial solution to this presents the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method or Quasi-Newton method [188]. The authors propose to approximate $\mathbf{H}^{-1}$ iteratively by only using first order gradient information. However, the algorithm still needs to store a $N \times N$ matrix and the computational complexity is $O(N^2)$. Further, LeCun *et al.* [138] presented a method to iteratively estimate the largest eigenvalue of $\mathbf{H}$ and also showed that the optimal learning rate λ is approximately the inverse of this value.

2.8.6 Radial Basis Function Networks

Introduction

The idea behind Radial Basis Function (RBF) Networks, shortly RBF, is based on Cover's theorem [48]:

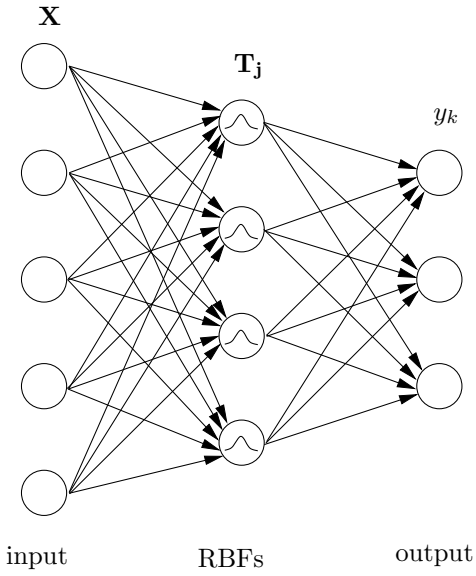


Figure 2.16: The architecture of a RBF Network

> "A complex pattern-classification problem cast in a high-dimensional space *nonlinearly* is more likely to be linearly separable than in a low-dimensional space."

Powell [187] introduced RBFs as a solution to the real multivariate interpolation problem but Broomhead and Lowe [30] were the first to exploit the use of RBFs in the context of NNs. They have been applied to many real-world problems since then, notably in the fields of functional approximation, time-series modeling and pattern classification.

RBF Networks are three-layer feed-forward NNs similar to MLPs, *i.e.* they consist of

- an *input* layer having no weights and receiving the input pattern,
- a *hidden* layer implementing the non-linear RBFs and fully connected to the input layer,
- an *output* layer consisting of perceptrons with linear activation functions.

Figure 2.16 illustrates the principal architecture of RBF Networks.

The principal difference to MLPs are the hidden neurons performing a non-linear mapping onto a subspace which is characterized by the RBFs that are parameterized by the respective neurons.

Thus, for a RBF Network having I inputs, J hidden neurons and K outputs, the outputs y_k are are calculated as follows:

$$y_k = \sum_{j=0}^{J} w_{kj} \Phi_j(||\mathbf{X} - \mathbf{T_j}||) \quad , \tag{2.78}$$

where w_{kj} is the weight from hidden neuron j to output neuron k, Φ_j is the RBF integrated by hidden neuron j with center $\mathbf{T_j} \in \mathbb{R}^I$, $\mathbf{X} \in \mathbb{R}^I$ is the input

vector, and $||\cdot||$ represents some norm, in general the L_1 or L_2 norm. Φ_0 can be set to 1 in order to use bias terms (*i.e.* w_{k0}). As can be noticed, the output of a hidden neuron j depends on the distance of $\mathbf{X}$ from the neuron's center $\mathbf{T_j}$, a distance of zero leading to a maximum or minimum activation according to the function Φ_j chosen.

The most common RBF is the Gaussian function:

$$\Phi_j(||\mathbf{X} - \mathbf{T_j}||) = \exp\left[-(\mathbf{X} - \mathbf{T_j})^T \Sigma_j^{-1} (\mathbf{X} - \mathbf{T_j})\right] \quad , \tag{2.79}$$

Σ_j being the covariance matrix which determines the shape, size and orientation of the *receptive field* of the neuron j, *i.e.* a subspace of the input space where Φ_j is having a higher activation. For example, for a diagonal covariance matrix with equal diagonal elements the receptive field is a hypersphere centered at $\mathbf{T_j}$; in the general case it is a hyperellipsoid.

Training

As MLPs, RBF Networks are trained using an labeled training data set and by adjusting its parameters such that an error function is minimized. In most of the cases, the error function is the sum over all training examples of the squared distances between the NN output and the desired output (see Eq. 2.52).

However, different training strategies can be adopted w.r.t. the hidden layer:

1. use fixed Gaussian RBFs chosen at random
2. use RBFs with fixed variance and adjust their centers by an unsupervised learning algorithm
3. adjust centers and variances of the RBFs by a supervised learning algorithm.

In the first case, the only parameters to learn are the weights and the biases of the output neurons. This can be done for example by the Least-Mean-Squares (LMS) algorithm or Error Backpropagation (*c.f.* Algorithm 3).

The second learning strategy consists of two stages. In the first stage, the centers of the RBFs are initialized randomly and then determined iteratively by a clustering algorithm, such as k-means, Linear Vector Quantization (LVQ) or Self-Organizing Maps (SOM). The variances are then computed using the variances of the respective clusters. In the second stage, the weights of the output neurons are calculated by a supervised learning algorithm.

The third learning strategy is completely supervised, *i.e.* the centers, the covariance matrix and the weights of the output neurons are adjusted iteratively by offline Backpropagation.

2.8.7 Self-Organizing Maps

Introduction

The NNs discussed so far are all used for supervised learning. Self-Organizing Maps (SOM) however rely on an *unsupervised learning* algorithm and are used when the structure of the training data is unknown. This approach has been proposed by Willshaw, von der Malsburg [258] and Kohonen [123]. The idea is to project high-dimensional input vectors onto a low-dimensional map (mostly 1D,

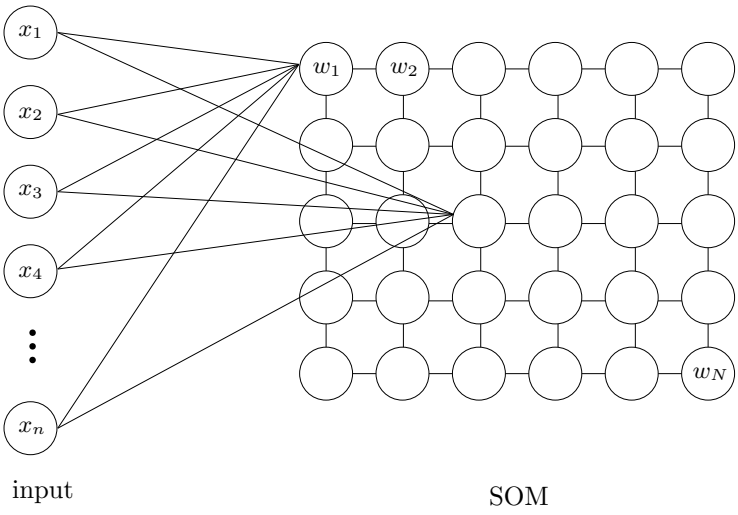


Figure 2.17: A two-dimensional SOM with rectangular topology

2D or 3D) such that the topology of the data (*i.e.* neighborhood relationships) is preserved and the input data is represented with optimal accuracy.

Thus, a SOM consists of a d-dimensional map of N neurons where each neuron i has a fixed position $\mathbf{p}_i \in \mathbb{N}^d$ which is used during training to define a spatial relationship between neighboring units. Further, each neuron contains a vector $\mathbf{w}_i \in \mathbb{R}^n$ that stores the coordinates it represents in the input space. Figure 2.17 illustrates a two-dimensional SOM with a rectangular topology. Then, an arbitrary input vector $\mathbf{v}$ is projected by identifying the Best Matching Unit (BMU) of the map, *i.e.* the neuron with the closest weight vector in terms of the Euclidean distance measure. Thus, we have

$$i_{\text{BMU}} = \underset{i}{\operatorname{argmin}} ||\mathbf{w}_i - \mathbf{v}|| \qquad i = 1..N \quad . \tag{2.80}$$

Training

As mentioned above, the training of SOMs is unsupervised. The algorithm is iterative, and we now define each parameter as a function of the iteration step (or time step) t. First, the weights $\mathbf{w}_i(0)$ of all neurons are initialized at random. Subsequently, at each iteration and for each training example $\mathbf{x}$, the BMU $i_{\text{BMU}}(t)$ according to the current weights $\mathbf{w}_i(t)$ is determined (Eq. 2.80). Each weight vector $\mathbf{w}_i(t)$ is then updated using the following formula:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \rho(t)\, h(||\mathbf{p}_i - \mathbf{p}_{i_{\text{BMU}}}||, t)\, (\mathbf{x} - \mathbf{w}_{i_{\text{BMU}}}) \quad , \tag{2.81}$$

where $h(||\mathbf{p}_i - \mathbf{p}_j||, t)$ is a function determining the influence of the weight update on the neighborhood of the BMU in the map. It is a symmetric function having its maximum at 0 and monotonically decreasing towards 0 with increasing distance $||\mathbf{p}_i - \mathbf{p}_j||$. A common choice for h is a Gaussian function:

$$h(||\mathbf{p}_i - \mathbf{p}_j||, t) = \exp\left(-\frac{||\mathbf{p}_i - \mathbf{p}_j||}{2\,\sigma(t)^2}\right) \tag{2.82}$$

In Eq. 2.81, the Gaussian function is centered at $\mathbf{p}_{i_{\text{BMU}}}$. Thus, the BMU receives the largest update, and it is smaller for the neighboring units according to their distance to the BMU in the map. The influence of the neighborhood is gradually narrowed by decreasing the variance $\sigma(t)$. For example:

$$\sigma(t) = \sigma(0)\ e^{-\frac{t}{\tau_1}} \quad , \tag{2.83}$$

where $\sigma(0)$ is the initial variance and τ_1 is some positive constant.

The function $\rho(t) \in \mathbb{R}$ is the learning rate and monotonically decreasing towards 0 with increasing time t. A popular choice is:

$$\rho(t) = \rho(0)\ e^{-\frac{t}{\tau_2}} \quad , \tag{2.84}$$

where $\rho(0)$ is the initial learning rate and τ_2 is some positive constant.

Algorithm 7 summarizes the self-organizing map training with J training examples.

Algorithm 7 A training algorithm for Self-Organizing Maps

$t = 0$
Initialize the weights $\mathbf{w}_i$ at random $\quad (\forall\, i = 1..N)$
for $t = 0$ to $t_{\max}$ **do**
  $\sigma(t) = \sigma(0)\ \exp(-t/\tau_1)$
  $\rho(t) = \rho(0)\ \exp(-t/\tau_2)$
  for $j = 0$ to J **do**
    $i_{\text{BMU}} = \text{argmin}_i\ ||\mathbf{w}_i - \mathbf{x}_j|| \quad (\forall\, i = 1..N)$
    for $i = 1$ to N **do**
      $\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \rho(t)\ h(||\mathbf{p}_i - \mathbf{p}_{i_{\text{BMU}}}||, t)\ (\mathbf{x}_j - \mathbf{w}_{i_{\text{BMU}}})$
    end for
  end for
end for

The initialization of σ and ρ with a relatively high value and their gradual decrease is important for the self-organization process. At the beginning, weight updates have an effect on almost the entire map and lead to an ordering of the neurons. With increasing time, the SOM gradually converges to a stable solution.

Figure 2.18 shows the evolution of a two-dimensional SOM of size 25×25 after 0, 20, 100, 1000, 5000 and 100000 training iterations [124]. The 10000 vectors of the training set are 2D vectors that are randomly drawn from the unit square. The initial weights are close to the center of the unit square as can be seen in the top left of Fig. 2.18.

2.9 Conclusion

In this chapter, we presented the most important machine learning approaches used for object detection and recognition in images. We began with some basic statistical projection methods which map the input images onto a lower-dimensional sub-space in order to select the most prominent features for further processing. Then, we focused on a statistical projection method, known as Active Appearance Model, which is more specific to facial analysis. By modeling

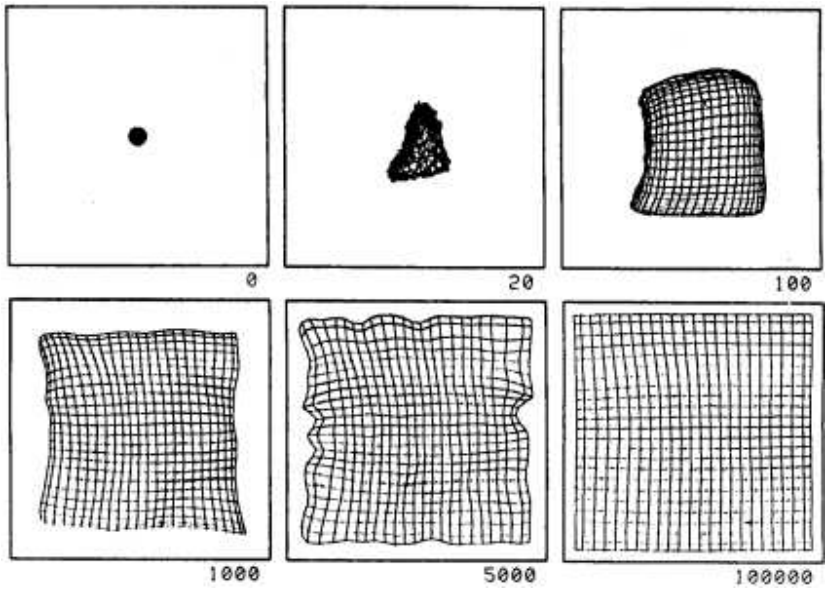


Figure 2.18: Evolution of a two-dimensional SOM during training

explicitely and separately shape and texture of the objects being processed, this technique allows to cope with varying pose and local deformations, *e.g.* facial expressions.

In the succeeding section, we described a rather different approach, called Hidden Markov Models (HMM), which focuses on the sequential aspect of the data and models it in a probabilistic way. Being initially designed for one-dimensional signals, *e.g.* speech, we outlined some existing methods extending HMMs for two-dimensional data such as images.

Another machine learning approach for visual object detection and recognition tasks is called Adaptive Boosting, short Adaboost. The main idea of this technique is to combine many weak classifiers based on simple features into a strong, more complex classifier by means of a specific selection and weighting procedure iteratively focusing on more and more difficult training examples.

Then, we briefly described Support Vector Machines, which are binary linear classifiers trained not only to minimize the empirical error on the training data but also the so-called structural risk, *i.e.* the generalization capacity of the resulting classifier. We also describe how this approach can be extended to the non-linear case and to multiple classes.

A more recent approach which is much more specific to image classification and object recognition is called "Bag of Local Signatures". It is based on the extraction of salient feature points in images and a subsequent invariant description of the respective local image patches. The quantization of the resulting local patches and the construction of histograms of these patches over the whole images then allows a classification of the visual content of the respective images.

Finally, we described a classical but nevertheless very versatile machine learning approach, known as Neural Networks. Starting with a description of the earliest models, such as the Perceptron and the Multi-Layer Perceptron,

and the way they can be trained we then outlined some more specific neural approaches being used for image processing tasks, *i.e.* Auto-Associative Neural Networks, Radial Basis Function Networks and Self-Organizing Maps.

A neural network approach showing to be particularly appropriate for visual object detection and recognition tasks and being inspired by biological evidence in the visual system of mammals is called *Convolutional Neural Network*. Being the foundation of the methods proposed in this work, we will describe this approach in more detail in the following chapter.

Chapter 3

Convolutional Neural Networks

3.1 Introduction

Multi-layer feed-forward Neural Networks (NN) have shown to be a very powerful machine learning technique as they can be trained to approximate complex non-linear functions from high-dimensional input examples. Classically, standard Multi-Layer Perceptrons (MLP) (*c.f.* section 2.8.3) have been utilized in pattern recognition systems to classify signatures coming from a *separate* feature extraction algorithm operating on the input data. However, the manual choice of the feature extraction algorithm and the features to classify is often empirical and therefore sub-optimal. Thus, a possible solution would be to directly apply the NN on the "raw" input data and let the training algorithm, *e.g.* Backpropagation, find the best feature extractors by adjusting the weights accordingly.

The problem with this approach is that when the input dimension is high, as in images, the number of connections, thus the number of free parameters is also high because each hidden unit would be fully connected to the input layer. Typically, this number may be in the order of several 10,000 or rather several 100,000 according to the application. The number of training examples, however, might be relatively small compared to the pattern dimension, which means that the NN would have a too high complexity and, thus, would tend to overfit the data.

Another disadvantage of this type of MLP comes from the fact that its input layer has a fixed size and the input patterns have to be presented well aligned and/or normalized to this input window, which, in practice, is a rather complicated task. Thus, there is no built-in invariance w.r.t. small translations and local distortions.

Finally, fully-connected NN architectures do not take into account correlations of neighboring input data. However, in pattern recognition problems there is generally a high amount of local correlation. Thus, it would be preferable to extract local features and combine them subsequently in order to perform the detection or recognition.

As we will see in this chapter, Convolutional Neural Networks (CNN) are an

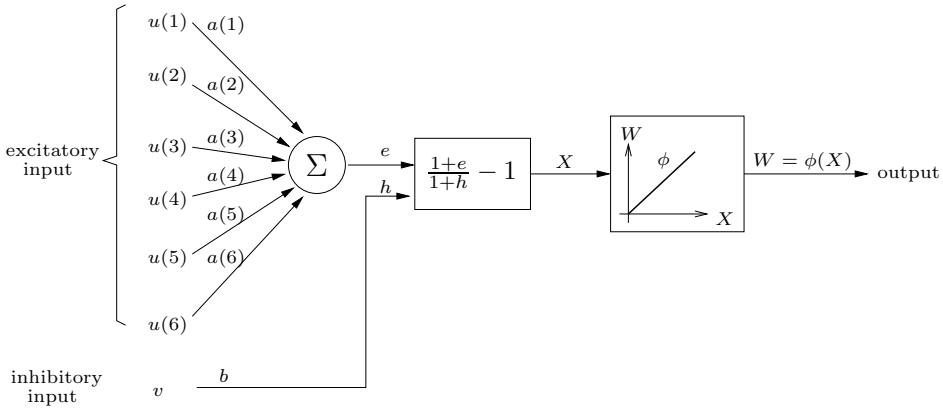


Figure 3.1: The model of a S-cell used in the Neocognitron

approach that tries to alleviate the above mentioned problems. That is, they automatically learn local feature extractors, they are invariant to small translations and distortions in the input pattern, and they implement the principle of weight sharing which drastically reduces the number of free parameters and thus increases their generalization capacity compared to NN architectures without this property.

3.2 Background

3.2.1 Neocognitron

The first implementation of a CNN was the so-called *Neocognitron* proposed by Fukushima [74, 75, 76, 77] which has been originally applied to the problem of handwritten digit recognition. The Neocognitron makes use of receptive fields, *i.e.* each neuron is only connected to a sub-region corresponding to a certain number of neighboring neurons, in the preceding layer. This idea has been inspired by the discovery of locally-sensitive, orientation-selective neurons in the cat's visual system by Hubel and Wiesel [106]. Local connections have been used many times with NNs [73, 133, 167]. They can be used to detect elementary visual features in images, such as oriented edges, end points or corners. As one feature extractor can be useful in several parts of the input image, the weights are forced to be identical for all possible locations of the receptive field, a principal called *weight sharing* [208].

The outputs of the respective neurons with identical weights, the so-called *S-cells*, form a *S-plane*. Fig. 3.1 illustrates the model of a S-cell. Note that besides the input from the receptive field, called *excitatory* input e, there is also an *inhibitory* input h which has a negative effect on the activation of the neuron. If h is greater than e the output of the neuron is zero. The excitatory input is calculated as follows:

$$e = \sum_{i=1}^{N} a(i)\, u(i) \quad , \tag{3.1}$$

where $a(i)$ are the training weights, $u(i)$ denote the inputs from the preceding cells, and N is the number of weights. For the inhibitory input, we have:

$$h = b\, v \quad , \tag{3.2}$$

where b is a trainable weight. In the Neocognitron, the input v is calculated as the weighted root mean-squared values coming from the receptive field and thus represents some kind of normalization. Finally, the activation of a S-cell is:

$$u_S(i) = \phi\left(\frac{1+e}{1+h} - 1\right) \quad , \tag{3.3}$$

where the activation function ϕ is defined as:

$$\phi(X) = \begin{cases} X & \text{if } X \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Fukushima [76] showed that the outputs of the S-cells approximate a convolution normalized by the length of the weight vector and the input vector. Several S-planes, each containing a different set of weights, can be used to extract different features at the same locations. The set of S-planes at one particular level form a *S-layer*.

The exact position of each feature is not very important in most cases. Therefore, each S-plane is followed by a *C-plane* which reduce the resolution of the respective S-plane by a constant factor, *e.g.* two and thus performs a kind of sub-sampling or blurring. The sub-sampling also reduces the sensitivity of the NN to small shifts and distortions of the input pattern.

By alternating C-layers and S-layers and combining the outputs of the respective maps, one can construct more complex feature extractors. This principle was inspired by Hubel and Wiesel's notion of "simple" and "complex" cells and it has been implemented in the Neocognitron where the first S-planes extract simple visual features, such as oriented edges or corner points, and the following layers combine them to extract more complex features, such as combinations of line segments.

The topology of the basic Neocognitron is illustrated in Fig. 3.2. It has an input layer u_{C0} and four alternating S- and C-layers, $u_{S1}, u_{C1}, \ldots, u_{S4}, u_{C4}$, where u_{C4} is the output layer consisting of single neurons representing the decision of the NN for some input pattern, in this case a digit. Note that the inhibitory inputs v are not depicted in Fig. 3.2. In fact, all inhibitory inputs of the S-planes of one layer u_{Si} are connected to a so-called *V-plane* u_{Vi}. The V-neurons of each V-plane receive their inputs from corresponding sub-regions in each preceding C-plane. The output is the weighted root mean square of these values and form the input v of the S-cells at the respective positions.

The training of the Neocognitron can be supervised or unsupervised. The first implementations used an unsupervised training algorithm based on self-organization and a particular reinforcement learning rule. However, later studies showed that the performance of the Neocognitron trained with a supervised algorithm was higher than with the original self-organization method. Further, training is performed layer by layer, *i.e.* starting from the first S-layer u_{S1} until the output layer u_{C4}, such that the training of a particular layer is only started after the training of the previous layer is finished. A reinforcement technique is

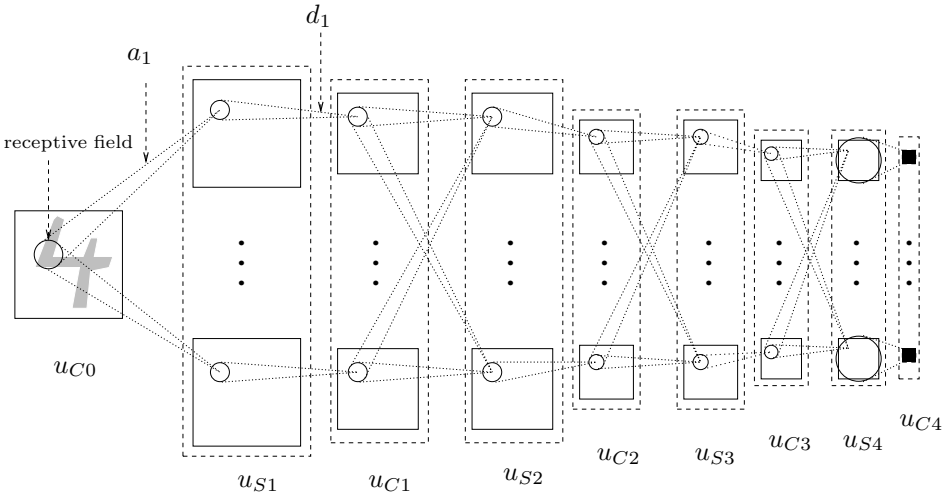


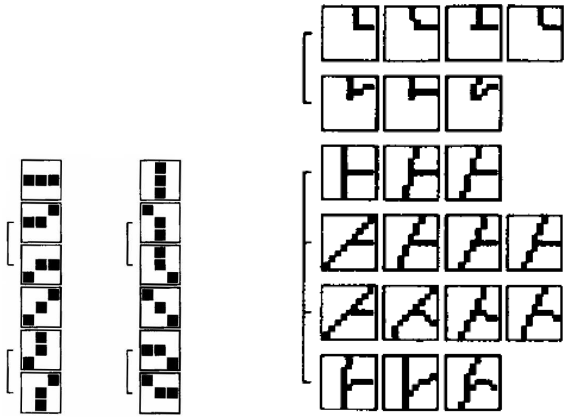
Figure 3.2: The topology of the basic Neocognitron

employed by the supervised algorithm in order to train each plane in a layer to detect a specific visual feature, *e.g.* a horizontal edge. The trainable parameters are the weights a and b, whereas the weights of the C-planes are not modified by the training algorithm and fixed at the beginning. At each step, the input pattern is propagated until the layer that is currently learned. For example, with handwritten digit recognition, the first layer is trained to recognize some very simple 3×3 pixel patterns, representing oriented edges (see Fig. 3.3(a)). The second layer is then trained to detect some more complex feature, *e.g.* oriented line end points, corner points or combinations of oriented line segments (see Fig. 3.3(b)). Each S-plane learns a particular pattern. The following S-layers are trained to recognize more and more complex features until the final S-layer u_{S4}, the planes of which are supposed to recognize the whole digit patterns.

The complete description of the rather complex Neocognitron would be beyond the scope of this work, and we refer to the literature [73, 74, 75, 76, 77] for further details. Fukushima presented many extensions to the basic model described here. For example, a non-linear activation function ϕ or the concept of *selective attention* [75], where there are not only forward but also backward connections of the layers and also interactions between the different planes of a layer controlling the learning process.

3.2.2 LeCun's Convolutional Neural Network model

An important breakthrough of CNNs came with the widespread use of the Backpropagation learning algorithm (*c.f.* section 2.8.5) for multi-layer feed-forward NNs. LeCun *et al.* [134] presented the first CNN that was trained by Backpropagation and applied it to the problem of handwritten digit recognition. As this model forms the basis of all the NN architectures presented in this work we will describe it in more detail in this section. From now on, when we employ the term *Convolutional Neural Network* we principally refer to NN models that are similar to the one proposed by LeCun *et al.* [134], which is actually a simpler model than the Neocognitron and its extensions.



(a) Layer u_{S1} (b) Layer u_{S2}

Figure 3.3: Some training examples used to train the first two S-layers of Fukushima's Neocognitron. Right: each row represents the training examples of *one* specific S-plane

Nevertheless, the architecture of LeCun's CNN and Fukushima's Neocognitron resemble in many aspects. It is a sequence of alternating S-layers and C-layers, here called convolution and sub-sampling layers, consisting of so-called *feature maps* corresponding to the S- and C-planes of the Neocognitron. Note that, contrary to what one would expect from the notations, S-layers correspond to convolution layers and C-layers to sub-sampling layers. The CNN model also implements the concepts of receptive fields and weight sharing. However, the model of the individual neurons is the basic Perceptron [200] with a sigmoid activation function. That means, there is no inhibitory input and no V-planes, which simplifies the overall model and architecture. A further advantage of this model is that the network does not have to be trained layer by layer but all the weights are adjusted iteratively by Error Backpropagation minimizing a global error function. As a consequence, the features to extract, *i.e.* edges, corners etc., are not chosen manually but found automatically by learning the convolution masks formed by the weights of respective feature maps (see section 3.3).

Figure 3.4 shows the topology of the CNN proposed by LeCun *et al.* [134], in later work referred to as *LeNet-1*. The input layer $y^{(0)}$ is of size 28×28 and receives the gray-level image containing the digit to recognize. The pixel intensities are normalized between -1 and $+1$. The first hidden layer H1 consists of four feature maps $y_j^{(1)}$ each having 25 weights $w_{j0}^{(1)}(u,v)$, constituting a 5×5 trainable kernel, and a bias $b_j^{(1)}$. Thus, the values of the feature maps $y_j^{(1)}(x,y)$ are obtained by convolving the input map $y^{(0)}$ with the respective kernel $w_{j0}^{(1)}$ and applying an activation function $\phi^{(1)}$ to the result:

$$y_j^{(1)}(x,y) = \phi^{(1)}\Big(\sum_{(u,v)\in K} w_{j0}^{(1)}(u,v)\, y^{(0)}(x+u, y+v) \; + b_j^{(1)} \Big) \,, \tag{3.5}$$

where $K = \{(u,v) \in \mathbb{N}^2 \,|\, 0 \le u < 5 \text{ and } 0 \le v < 5\}$ Note that due to border effects the resulting feature maps are smaller, *i.e.* 24×24.

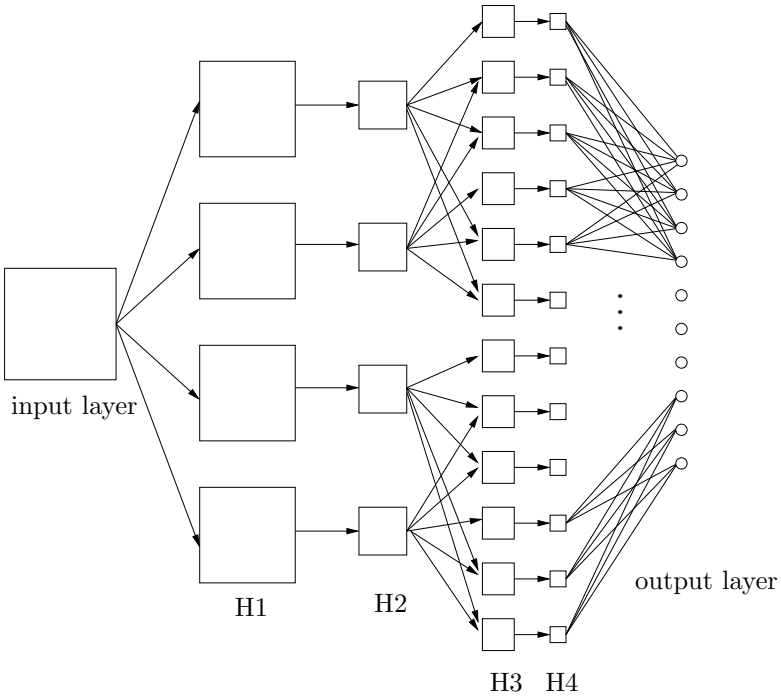


Figure 3.4: The architecture of the CNN proposed by LeCun *et al.* [134] for handwritten digit recognition (LeNet-1)

Each convolution map is followed by a sub-sampling map $y_j^{(2)}$ which performs a kind of averaging and reduces the dimension of the respective convolution map by a factor two. Hence, the sub-sampling maps of layer H2 are of size 12×12. Further, each sub-sampling map has a weight $w_j^{(2)}$ and a bias $b_j^{(2)}$. Thus,

$$y_j^{(2)}(x,y) = \phi^{(2)}\Big(w_j^{(2)} \times \sum_{(u,v)\in\{0,1\}^2} y_j^{(1)}(2x+u, 2y+v) + b_j^{(2)}\Big). \tag{3.6}$$

$\phi^{(2)}$ again is an activation function which has to be continuous and differentiable as required by the Backpropagation learning algorithm.

Figure 3.5 illustrates the process of convolution and sub-sampling.

Layers H3 and H4, each contain 12 convolution and sub-sampling maps of dimensions 8×8 and 4×4 respectively. The function they implement is exactly

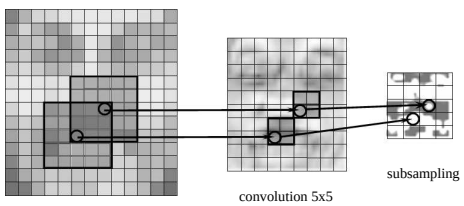


Figure 3.5: An input image followed by a feature map performing a 5×5 convolution and a 2×2 sub-sampling map

the same as the layers H1 and H2 except that the feature maps of layer H3 perform a 3×3 convolution instead of a 5×5. Further, one convolution map j in layer H3 can have several kernels $w_{ji}^{(3)}$ operating on different maps i in the preceding layer H2. The activation of respective convolution maps is then simply the sum of all convolution results and the bias. Thus, the general activation formula for a convolution map j in layer l is:

$$y_j^{(l)}(x,y) = \phi^{(l)}\Big(\sum_{i \in I} \sum_{(u,v) \in K} w_{ji}^{(l)}(u,v)\; y_i^{(l-1)}(x+u, y+v) \; + b_j^{(l)}\Big)\,, \qquad (3.7)$$

where $K = \{(u,v) \in \mathbb{N}^2 \,|\, 0 \leq u < s_x\,;\, 0 \leq v < s_y\}$, (s_x, s_y) is the dimension of the convolution kernel, and I is the set of maps of the preceding layer the convolution map j is connected to. The feature maps in layer H3 learn to extract more complex features by combining the extracted simple features of layer H2. The connection scheme is illustrated in Fig. 3.4.

Finally, the output layer contains a set of 10 neurons, fully connected to the previous sub-sampling maps of layer H4, and representing the 10 digits to recognize. The "winning" neuron is supposed to respond with the value $+1$ and the other neurons with -1.

In total, the network has 4635 units, 98442 connections, but, due to weight sharing, only 2578 independent parameters to learn.

3.3 Training Convolutional Neural Networks

The training of CNNs is very similar to the training of other types of NNs, such as ordinary MLPs. A set of training examples is required, and it is preferable to have a separate validation set in order to perform cross-validation and "early stopping" and to avoid overtraining (see section 2.8.5).

To improve generalization, small transformations, such as shift and distortion, can be manually applied to the training set. Consequently the set is augmented by examples that are artificial but still form valid representations of the respective object to recognize. In this way, the CNN learns to be invariant to these types of transformations.

In terms of the training algorithm, in general, online Error Backpropagation leads to the best performance of the resulting CNN. Therefore, this algorithm has been applied for all experiments throughout this work, and it will be described in detail in the following section. In section 3.3.2, we will also outline some alternatives published in the literature.

3.3.1 Error Backpropagation with Convolutional Neural Networks

As the online Backpropagation algorithm is the most commonly used learning algorithm for CNNs it will be described in more detail in this section. In fact, it is almost identical to the Backpropagation algorithm for standard MLPs (see section 2.8.5). The only difference to take into account is the weight sharing in the convolution and sub-sampling layers.

We also want to minimize the error function E_p after each training example p:

$$E_p = \frac{1}{2}||\mathbf{o_p} - \mathbf{t_p}||^2 = \frac{1}{2}\sum_{k=1}^{K}(o_{pk} - t_{pk})^2 \quad , \tag{3.8}$$

where K is the number of output units, o_{pk} is the output of neuron k for pattern p and t_{pk} is the respective target value ($t_{pk} \in [-1, +1]$ in most cases). The learning process is an iterative procedure, where at each iteration the weight update is a small step in the opposite direction of the steepest gradient ∇E. Thus,

$$w_{ji}^{(l)} \quad \leftarrow \quad w_{ji}^{(l)} + \Delta w_{ji}^{(l)} = w_{ji}^{(l)} - \lambda \frac{\partial E_p}{\partial w_{ji}^{(l)}} \quad , \tag{3.9}$$

where λ is the learning rate and $w_{ji}^{(l)}$ denotes the weight from neuron i to neuron j in layer l. For the sake of clarity, we will drop the pattern index p.

For the output layer we can adopt the equations of the MLP (see section 2.8.5):

$$\Delta w_{kj}^{(l)} = -\lambda \, \delta_k^{(l)} \, y_j^{(l-1)} \quad , \tag{3.10}$$

where

$$y_j^{(l-1)} \qquad \text{is the activation of neuron } j \text{ in layer } l-1, \tag{3.11}$$

$$\delta_k^{(l)} = e_k \, \phi'(V_k^{(l)}) \qquad \text{is the local gradient,} \tag{3.12}$$

$$e_k = o_k - t_k \qquad \text{denotes the error, and} \tag{3.13}$$

$$V_k^{(l)} = \sum_j w_{kj}^{(l)} \, y_j^{(l-1)} \qquad \text{is the weighted sum of all inputs } y_j^{(l-1)} \text{ of neuron } k. \tag{3.14}$$

The activation function ϕ of the output neurons is usually the hyperbolic tangent function:

$$\phi(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad . \tag{3.15}$$

If there is an additional hidden layer containing simple neurons between the last sub-sampling layer and the output layer the equation is the same as for MLPs, and we have:

$$\delta_j^{(l)} = e_j^{(l)} \, \phi' \left(V_j^{(l)} \right) = \left(\sum_{k=1}^{K} \delta_k^{(l+1)} w_{kj}^{(l+1)} \right) \phi' \left(V_j^{(l)} \right) \tag{3.16}$$

for the local gradient of hidden neuron j. We denote $e_j^{(l)}$ the error at neuron j back-propagated from the neurons of the following layer $l+1$.

Now, let us consider the case of a *convolution layer*. The update of a particular weight $w_{ji}(u, v)$ from feature map i to j at kernel position (u, v) then actually becomes a sum over all positions (x, y) of the feature map.

$$\Delta w_{ji}^{(l)}(u, v) = -\lambda \sum_{(x,y)} \left(\delta_j^{(l)}(x, y) \, y_i^{(l-1)}(x + u, y + v) \right) \tag{3.17}$$

Figure 3.6 illustrates the error Backpropagation with convolution maps and the relation of the different variables, i, j, u, v, x and y. The bias $b_j^{(l)}$ of convolution

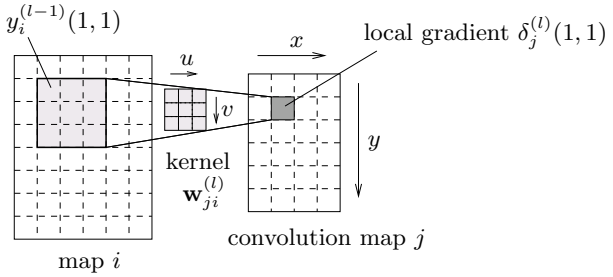


Figure 3.6: Error Backpropagation with convolution maps

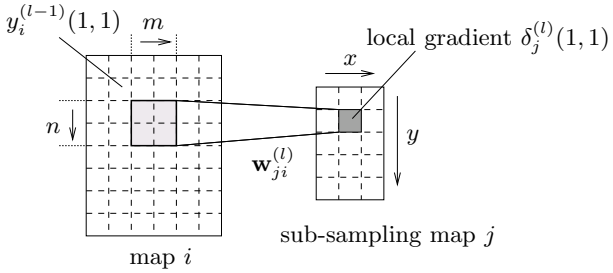


Figure 3.7: Error Backpropagation with sub-sampling maps

map j in layer l is simply updated by adding the following term:

$$\Delta b_j^{(l)} = -\lambda \sum_{(x,y)} \delta_j^{(l)}(x,y) \tag{3.18}$$

With *sub-sampling maps* the calculation is slightly different. There is only one weight $w_{ji}^{(l)}$ for each connection from map i to sub-sampling map j. Note that usually, there is simply a one-to-one connection from convolution to sub-sampling maps, thus $i = j$. For the general case of a weight $w_{ji}^{(l)}$ from map i in layer $l-1$ to a sub-sampling map j in layer l with the sub-sampling factors (or window-size) s_x and s_y, we have:

$$\Delta w_{ji}^{(l)} = -\lambda \sum_{(x,y)} \delta_j^{(l)}(x,y) \sum_{m=1}^{s_x} \sum_{n=1}^{s_y} y_i^{(l-1)}(xs_x + m\,,\, ys_y + n) \tag{3.19}$$

Figure 3.7 illustrates the relation of the respective variables. The bias update formula is the same as for convolution maps (Eq. 3.18).

The calculation of the local gradient $\delta_j^{(l)}$ of a map j in layer l depends on the type of succeeding layer $l+1$. If the layer $l+1$ is a *neuron layer*, we have:

$$\delta_j^{(l)}(x,y) = \sum_{k=1}^{K} \sum_{(x,y)} \delta_k^{(l+1)} w_{kj}^{(l+1)}(x,y) \quad , \tag{3.20}$$

which is an generalization of Eq. 3.16 taking into account not only the connection of single neurons with neurons but also the connection of a *map* with neurons $(x, y > 1)$. K is the number of neurons in layer $l+1$.

If the succeeding layer is a *convolution layer*, we have:

$$\delta_j^{(l)}(x,y) = \sum_{k \in K_c} \sum_{(u,v)} \delta_k^{(l+1)}(x,y) \cdot w_{kj}^{(l+1)}(u,v) \quad , \tag{3.21}$$

where K_c denotes the set of maps in layer $l+1$ connected to the convolution map j in layer l. It has to be taken into account that the convolution mask cannot cross the border of layer l. Thus, the respective terms are not included in the above sum.

In the case of layer $l+1$ being a *sub-sampling layer*, the local gradient is the following:

$$\delta_j^{(l)}(x,y) = \sum_{k \in K_s} \delta_k^{(l+1)}(\lfloor x/s_x \rfloor, \lfloor y/s_y \rfloor) \cdot w_{kj}^{(l+1)} \tag{3.22}$$

where s_x, s_y are the sub-sampling factors, K_s denotes the set of maps in layer $l+1$ connected to the sub-sampling map j in layer l and $\lfloor \cdot \rfloor$ is the floor function.

Algorithm 8 summarizes the online Backpropagation algorithm for CNNs composed of several alternating convolution and sub-sampling layers followed by one or more neuron layers.

3.3.2 Other training algorithms proposed in the literature

Although, Error Backpropagation is the most commonly used training algorithm for CNNs, some alternatives have been proposed in the literature. In principle, most of the learning algorithms for MLPs can be used for CNNs too if one takes into account weight sharing. However, in practice, the standard online Backpropagation algorithms outperforms these methods in most applications. Nevertheless, we will outline here some of the alternatives presented in the literature and briefly explain how they have been applied to CNNs.

Extending the Neocognitron learning algorithm where each layer is trained separately, Neubauer [170] presented a hybrid supervised-unsupervised approach where the features to recognize do not have to be defined manually. Here, the parameters of the first hidden layer are trained by randomly extracting image regions of the size of the receptive field from the training images and then performing a PCA on these image patches or training a 3-layer auto-encoding NN [208, 45]. The same procedure is repeated for each following hidden layer where the training image patches are first propagated until the previously trained layer, and then the respective output patches of this layer are used as an input for PCA or auto-encoding. This type of learning algorithm can be preferable when it is not clear which features should be extracted for a given task, for example face recognition. After having trained all the hidden layer, the weights of the final neuron layer are determined by a supervised algorithm, *e.g.* LMS fitting.

Li *et al.* [142] used an extension of the standard Backpropagation algorithm, called *Delta-Bar-Delta* algorithm, to train a CNN for text detection in images. It adjusts the learning rate for each weight separately according to the evolution of respective gradients. Thus, at the beginning all learning rates λ_i of weights w_i are initialized to some small value, and at each iteration they are updated

Algorithm 8 The online Backpropagation algorithm for Convolutional Neural Networks

Initialize all weights of the CNN to some small random value
Set the learning rates $\lambda^{(l)}$ to small positive values
$n = 1$
repeat
  for $p = 1$ to P **do**
    propagate pattern $\mathbf{x_p}$ through the network
    for $k = 1$ to K **do**
      $\delta_k^{(L)} = (o_k - t_k)$
    end for
    for layers $l = L - 1$ to 1 **do**
      for maps $j = 1$ to J **do**
        for all positions (x, y) **do**
          if layer $l + 1$ is a neuron layer **then**
            $\delta_j^{(l)}(x,y) = \left(\sum_{k=1}^{K} \sum_{(x,y)} \delta_k^{(l+1)} w_{kj}^{(l+1)}(x,y)\right) \phi'(V_k^{(l)})$
          else if layer $l + 1$ is a convolution layer **then**
            $\delta_j^{(l)}(x,y) = \left(\sum_{k \in K_c} \sum_{(u,v)} \delta_k^{(l+1)}(x,y) \cdot w_{kj}^{(l+1)}(u,v)\right) \phi'(V_k^{(l)})$
          else if layer $l + 1$ is a sub-sampling layer **then**
            $\delta_j^{(l)}(x,y) = \left(\sum_{k \in K_s} \delta_k^{(l+1)}(\lfloor x/s_x \rfloor, \lfloor y/s_y \rfloor) \cdot w_{kj}^{(l+1)}\right) \phi'(V_k^{(l)})$
          end if
        end for
      end for
    end for
    for $l = 1$ to L **do**
      for $j = 1$ to J **do**
        for all weights of map/neuron j **do**
          if layer l is a convolution layer **then**
            $\Delta w_{ji}^{(l)}(u,v) = -\lambda^{(l)} \sum_{(x,y)} \left(\delta_j^{(l)}(x,y)\, y_i^{(l-1)}(x+u, y+v)\right)$
          else if layer l is a sub-sampling layer **then**
            $\Delta w_{ji}^{(l)} = -\lambda^{(l)} \sum_{(x,y)} \delta_j^{(l)}(x,y) \sum_{m=1}^{s_x} \sum_{n=1}^{s_y} y_i^{(l-1)}(x s_x + m\,,\, y s_y + n)$
          else if layer l is a neuron layer **then**
            $\Delta w_{ji}^{(l)} = -\lambda^{(l)}\, \delta_j^{(l)}\, y_i^{(l-1)}$
          end if
          update weight: $w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} + \Delta w_{ji}^{(l)}$
        end for
        if layer l is a convolution or sub-sampling layer **then**
          $\Delta b_j^{(l)} = -\lambda^{(l)} \sum_{(x,y)} \delta_j^{(l)}(x,y)$
        else
          bias of neurons is already included in the weight vector
        end if
        update bias: $b_j^{(l)} \leftarrow b_j^{(l)} + \Delta b_j^{(l)}$
      end for
    end for
    $n = n + 1$
  end for
  $E = \frac{1}{2} \sum_{p=1}^{P} ||\mathbf{o_p} - \mathbf{t_p}||^2$
until $E < \epsilon$ or $n > max_{\text{iter}}$

as follows:

$$\lambda_i(n) = \lambda_i(n-1) + u \qquad \text{if } \frac{\partial E}{\partial w_i}(n) \cdot \bar{\delta}_i(n-1) > 0 \tag{3.23}$$

$$\lambda_i(n) = \lambda_i(n-1) \cdot d \qquad \text{if } \frac{\partial E}{\partial w_i}(n) \cdot \bar{\delta}_i(n-1) < 0 \tag{3.24}$$

$$\lambda_i(n) = \lambda_i(n-1) \qquad \text{otherwise} \quad , \tag{3.25}$$

where δ_i denotes the exponential averaged gradient w.r.t. weight w_i:

$$\bar{\delta}_i(n) = (1-\phi) \cdot \frac{\partial E}{\partial w_i}(n) \; + \; \phi \cdot \bar{\delta}_i(n-1) \quad , \tag{3.26}$$

with $\phi \in [0, 1]$. u and d are small positive constants, *e.g.* $u = 0.1$ and $d = 0.9$. The weights are then updated as with the standard Backpropagation algorithm:

$$w_i(n) = w_i(n-1) - \lambda_i(n) \cdot \frac{\partial E}{\partial w_i}(n) \quad . \tag{3.27}$$

Li *et al.* [142] showed that the Delta-Bar-Delta algorithm converges faster than Backpropagation with fixed learning rates and momentum term. However, the performance in terms of the detection rate of the CNN trained with the Delta-Bar-Delta algorithm is slightly inferior.

Ouellette *et al.* [178] proposed a learning method based on Genetic Algorithms (GA) and applied it to a CNN used to detect cracks in sewer pipes. Their motivation for using a GA instead of Backpropagation is the common assumption that GAs are less sensitive to getting stuck in local minima during the learning process. The idea of GAs is based on evolutionary concepts, *i.e.* a certain number of possible solutions, *i.e.* weight configurations, forming the so-called population is simultaneously maintained. Each solution, called chromosome, has to be encoded in a string of symbols, in this case a sequence of bits representing the integer weight values from 0 to 255 in binary form (from -127 to 128 for biases). Before training starts, the population is initialized with random chromosomes. Then, at each iteration, the following steps are repeated until a global evaluation criteria is satisfied:

1. **Fitness evaluation:** The fitness of each member is evaluated, *i.e.* the performance of the CNN on the training set with the respective weight configuration.

2. **Selection:** A subset of members, the parents, with the highest fitness value is selected to produce the offspring for the next generation.

3. **Crossover:** The chromosomes of the selected parents are split into two at random positions and the resulting partial bit sequences are combined to form the offspring, *i.e.* the new weight configurations.

4. **Mutation and Creep:** 1% of the current population is selected for mutation and creeping. Mutation means that the respective chromosome is randomly changed within $\pm 1\%$. Creeping is a similar technique which scales a chromosome by a random factor between -1 and $+1$.

Although the authors claim that the proposed algorithm is simpler to implement than Backpropagation they could not notice a significant performance improvement in terms of accuracy for the given application.

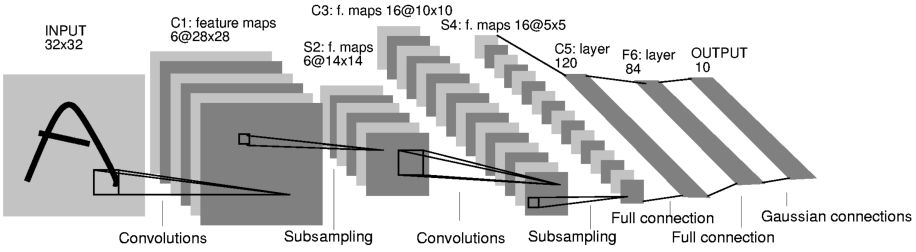


Figure 3.8: The architecture of LeNet-5

3.4 Extensions and variants

Many variants of the CNN model described in section 3.2.2 exist. Usually, the number of layers, the number of feature maps and their dimensions and the connection scheme are adapted to the given problem. Unfortunately, these parameters have to be determined experimentally as, up to the present, there is no algorithm to automatically determine the optimal architecture of a CNN for a given classification task. However, besides the architectural variants, there have been some major enhancements to the CNN model of LeCun *et al.* [134]. In the following, we will present the most important ones.

3.4.1 LeNet-5

A more complex CNN has been presented by LeCun *et al.* [136] and applied to handwritten character recognition. The authors called this architecture, *LeNet-5*. A larger training set has also been used to train this network improving its overall performance in terms of recognition rate as well as its generalization capacity. Figure 3.8 illustrates the principal architecture of LeNet-5.

It is composed of 7 layers, not counting the input layer. The input image is of size 32×32 pixels. As with LeNet-1, the first five layers, $C1, S2, C3, S4, C5$, are alternating convolution and sub-sampling layers with a 5×5 convolution mask and a sub-sampling factor of 2 respectively. The dimensions of the individual maps are noted in Fig. 3.8. The connections between the units of the respective layers is similar to LeNet-1, *i.e.* a full connection of the first layer to the input image and one-to-one connections in the sub-sampling layers. However, layer $C3$ follows a special non-symmetric connection scheme which is described in table 3.1. The convolution maps in $C5$ have a dimension of only $(1, 1)$ as the size of the maps in $S4$ is 5×5. Further, they are fully connected to each sub-sampling map leading to a high number of trainable parameters in this layer (*i.e.* 48,120).

Layer $F6$ is an additional hidden neuron layer, fully connected to layer $C5$ and composed of 84 units. The actual novel idea of LeNet-5 lies in the output layer which consists of fully connected Radial Basis Function (RBF) units, one for each character to recognize. To compute the outputs of a RBF unit y_i, we have:

$$y_i^{(6)} = \sum_j (x_j - w_{ji}^{(6)})^2 \ . \tag{3.28}$$

Thus, each RBF unit computes the squared Euclidean distance between its 84-dimensional input vector x_j and its weight vector $w_{ji}^{(6)}$. Further, the weight

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Table 3.1: The connection scheme of layer $C3$ of Lenet-5: a cross indicates a connection between a particular feature map in $S2$ (row) and a map in $C3$ (column)

vectors of layer $F6$, *i.e.* the centers of the RBFs, are fixed at the beginning in such a way that the 84 values are either -1 or $+1$ and form a 7×12 matrix representing a stylized image of the character. Thus, similar characters, like the uppercase "O", lowercase "o", and zero, will produce a similar output. This may be helpful for a linguistic post-processor working on the word level which could eliminate ambiguities due to confusions of similar characters.

An important point to note is the special error function used by LeCun *et al.* :

$$E(W) = \frac{1}{P} \sum_{p=1}^{P} \left(y^{(6)}_{pD_p} + \log(e^{-j} + \sum_i e^{-y^{(6)}_{pi}}) \right) , \tag{3.29}$$

where P is the number of training patterns, $y^{(6)}_{pi}$ is the activation of output neuron i when presenting pattern p, and D_p is the desired class of pattern p. Minimizing E means minimizing the two terms in the above equation. The first term, $y^{(6)}_{pD_p}$, represents the Euclidean distance of the desired output neuron to the 84-dimensional RBF center. Using only this term could lead to a trivial solution where all outputs are always zero. Hence, the second term has been introduced to also incorporate the outputs of the other neurons, and therefore it has a discriminative function. It ensures that the RBF centers are kept apart from each other and inhibits the trivial case where they "collapse" to the zero solution. The positive constant j prevents the error function from a further increase for classes that have already large outputs.

3.4.2 Space Displacement Neural Networks

CNNs have a fixed size input layer, also called retina, which can be used to detect or recognize particular objects in images or any other visual structure they have been trained to process. For example, a CNN trained to recognize handwritten characters requires the characters presented to the input to be size-normalized and more or less centered. However, there are applications where the size of the input is variable, such as the recognition of words or sentences or for detection tasks, like face detection, where the input images to process can be of arbitrary size.

A possible solution to this would be to pre-process the variable-size input image and segment it into different parts, *e.g.* the characters of a word or sentence. Unfortunately, there does not exist any reliable technique to perform

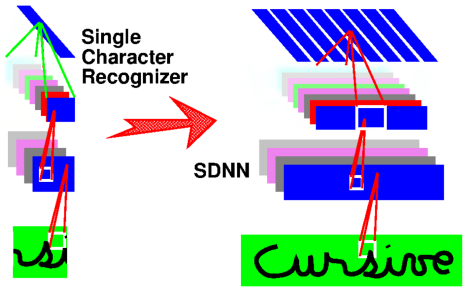


Figure 3.9: A Space Displacement Neural Network

this segmentation in most applications because of the too high variability of the pattern to recognize or detect. We therefore would have to place the retina at each location in the input image and propagate the respective values to obtain the output of the NN for that particular sub-region of the image. However, at two neighboring positions, there is a large amount of common computation as the sub-regions to process overlap and the convolution operations are the same in the overlapping parts. To take advantage of this, we can actually increase the size of the retina and all the convolution and sub-sampling layers accordingly. Thus, the convolution and sub-sampling with the respective masks is performed on the whole input image. Consequently, the output layer has to be replicated over all possible locations, and it effectively becomes a convolution layer. This trick leads to an architecture called Space Displacement Neural Network (SDNN). Figure 3.9 illustrates this by means of a CNN for connected character recognition. The output of the SDNN represents a sequence of outputs of the respective simple CNN at each possible location. In order to perform the final recognition or detection, this sequence has to be interpreted further in a separate post-processing step because at neighboring positions of a present object the SDNN might give the same answer due to some translation invariance of the CNN. Thus, there has to be some sort of grouping yielding exactly one response for each object to detect or recognize. Further, when the object to recognize is composed of a sequence of adjacent single entities, as the words in character recognition, the output of the SDNN *between* two such entities (characters) might be rather arbitrary and has to be discarded.

A classical approach to post-process the output sequence of a SDNN is to use a HMM. Such hybrid solutions have been proposed for example in the context of handwritten multi-digit or word recognition [121, 156, 19].

3.4.3 Siamese CNNs

Siamese Neural Networks have first been presented by Bromley *et al.* [29] using Time Delay Neural Networks (TDNN) and applying them to the problem of signature verification, *i.e.* to verify the authenticity of signatures. This idea was then adopted by Chopra *et al.* [39] who used Siamese *CNNs* and employed them in the context of face verification. More precisely, the system receives two face images and has to decide if they belong to the same person or not.

Siamese NNs learn a non-linear similarity metric by repeatedly presenting pairs of positive and negative examples, *i.e.* pairs of examples belonging to

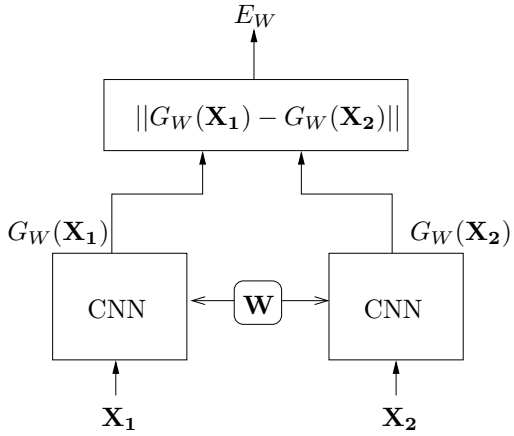


Figure 3.10: Illustration of a Siamese Convolutional Neural Network

the same class or not. The principal idea is to train the NN to map the input vectors into a non-linear subspace such that a simple distance, *e.g.* the Euclidean distance, in this subspace approximates the "semantic" distance in the input space. That means, two images of the same category are supposed to yield a small distance in this subspace and two images of a different category a large distance. Let us call this mapping $G_W(\mathbf{X})$ and its parameters (*i.e.* weights) $\mathbf{W}$. Thus, the goal is to learn the parameters $\mathbf{W}$ of the function $G_W(\mathbf{X})$ such that the similarity metric

$$E_W(\mathbf{X_1}, \mathbf{X_2}) = ||G_W(\mathbf{X_1}) - G_W(\mathbf{X_2})|| \tag{3.30}$$

is small if $\mathbf{X_1}$ and $\mathbf{X_2}$ belong to the same class and large if they belong to different classes. The choice of $G_W(\mathbf{X})$ is arbitrary and, in this case, is a CNN. Note that the parameters $\mathbf{W}$, *i.e.* the weights, are the same for both inputs, hence the name "siamese" NN, and therefore the distance metric is symmetric. Fig. 3.10 illustrates the functional scheme of this learning machine.

To make the system behave in the desired way, we need to define a global energy function, or loss function, $L(\mathbf{W})$ such that minimizing it decreases the distance between examples belonging to the same class (genuine pairs) and increases the distance between examples belonging to different classes (impostor pairs). A possible energy function would be:

$$\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{P} \left((1 - Y^i)\, L_G(E_W(\mathbf{X_1}, \mathbf{X_2})^i) + Y^i\, L_I(E_W(\mathbf{X_1}, \mathbf{X_2})^i) \right) , \tag{3.31}$$

where P is the number of image pairs, $(\mathbf{X_1}, \mathbf{X_2})^i$ is the i-th image pair and Y^i its label, *i.e.* 1 for a genuine and 0 for an impostor pair. L_G and L_I are the energy functions for genuine and impostor pairs respectively, where L_G is monotonically increasing and L_I is monotonically decreasing. Chopra *et al.* [39] require some more conditions to be satisfied, *e.g.* the existence of a margin, and they formalize some properties of L_G and L_I in order to ensure a correct behavior of the siamese CNN. We refer to [39] for more details on this.

The energy function for genuine pairs is usually defined as:

$$L_G = E_W^2 , \tag{3.32}$$

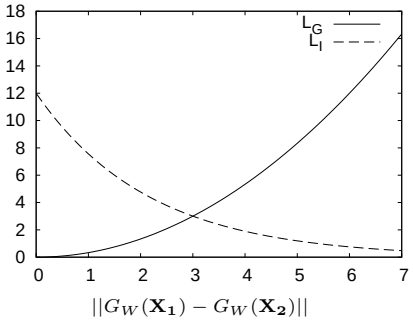


Figure 3.11: Example of positive (genuine) and negative (impostor) error functions for Siamese CNNs

i.e. we want the distance to be zero for pairs of examples belonging to the same category. However, the choice of L_I is more difficult. In general, we don't know the exact distance of an impostor pair, *e.g.* the face images of two different persons. A possible solution is to use:

$$L_I = e^{-tE_W} \tag{3.33}$$

with t being a positive constant. This prevents the projected vectors of two different classes from being too close. Chopra *et al.* [39] used the following global energy function:

$$\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{P} \left((1 - Y^i) \frac{2}{Q} \left(E_W(\mathbf{X_1}, \mathbf{X_2})^i \right)^2 + Y^i \, 2Qe^{-\frac{2.77}{Q} E_W(\mathbf{X_1}, \mathbf{X_2})^i} \right), \tag{3.34}$$

where the constant Q is the upper bound of E_W. Figure 3.11 shows the functions L_G and L_I for the one-dimensional case.

The architecture of the CNN in [39] is composed of 6 layers:

- C_1: a convolution layer with 15 feature maps and kernel size 7×7
- S_2: a sub-sampling layer containing 15 maps; sub-sampling window size: 2×2
- C_3: a convolution layer with 45 feature maps and kernel size 6×6; the connection scheme is similar to the one of LeNet-5 (see table 3.1)
- S_4: a sub-sampling layer containing 45 maps; sub-sampling window size: 4×3
- C_5: a fully connected convolution layer with 250 feature maps and kernel size 5×5
- F_6: a fully connected output layer containing 50 neurons.

The input layer of this CNN has the dimension 46×56.

Training is performed using a set of positive and negative examples, *i.e.* randomly selected genuine pairs and impostor pairs. At each training iteration, a positive pair and a negative pair is presented to the network. To update

the weights, the standard Backpropagation learning algorithm can be used (see Alg. 8 on page 57). To this end, the two examples of a pair $\mathbf{X_1}$ and $\mathbf{X_2}$ are successively input to the network and the error function is evaluated accordingly. Then, the errors with respect to $\mathbf{X_1}$, $\mathbf{X_2}$ or both are back-propagated layer by layer. Finally, the weights of the CNN are updated as detailed in section 3.3.1.

Once the network has been trained, a statistical analysis on the output vectors of the training images is performed, *i.e.* the mean feature vectors of each class and the variance-covariance matrix are computed. In this way, a multivariate Gaussian model is built for each subject. In order to verify the identity of a given test person, we can then compare the feature vector produced by the CNN with the statistical model of the claimed identity. A threshold has to be set which minimizes falsely accepted and falsely rejected images. For further details, see [39].

3.4.4 Shunting Inhibitory Convolutional Neural Networks

Shunting Inhibitory Convolutional Neural Networks (SICoNNets) represent a special type of CNN introduced by Tivive and Bouzerdoum [238] where the model of the neurons in the feature maps is different from the standard Perceptron model. It makes use of so-called *shunting inhibitory neurons*, a model inspired by neuro-physiological studies in the early 1960's. This model has then be applied to numerous visual information processing task [91, 27, 25, 28] and finally implemented in a feed-forward neural network structure by Bouzerdoum *et al.* [26]. The activation of shunting inhibitory neurons is governed by a differential equation, and the steady-state response of a neuron is given by a following formula which has been generalized by Arulampalam and Bouzerdoum [3]. Thus, for the activation z of a given neuron, we have:

$$z = \frac{g\left(\sum_{j=1}^{S_R} C_j I_j + b\right)}{a + f\left(\sum_{j=1}^{S_R} D_j I_j + d\right)} \,, \tag{3.35}$$

where I_j is the j-th input of the neuron, a is the passive decay rate, C_j and D_j are the excitatory and inhibitory weights of input j, b and d are constant biases, f and g are activation functions and S_R is the number of inputs from the 2D receptive field. Figure 3.12 illustrates the shunting inhibitory neuron model. In order to avoid division by zero the denominator of equation 3.35 is constrained to be positive:

$$a + f\left(\sum_{j=1}^{S_R} D_j I_j + d\right) > \epsilon \,, \tag{3.36}$$

where ϵ is a small positive constant.

The neuron model in the feature extraction layers is the major difference between a SICoNNet and a "traditional" CNN. SiCoNNets also have feature maps consisting of shunting inhibitory neurons, and they also implement weight sharing. Thus, the inputs I_j of a respective neuron of a feature map are coming from a two-dimensional receptive field in the previous map it is connected to. The weights C_j and D_j of a given feature map are then shared over all possible locations of the map. However, there is no explicit sub-sampling layer but it is integrated into the convolution layers by simply moving the receptive field

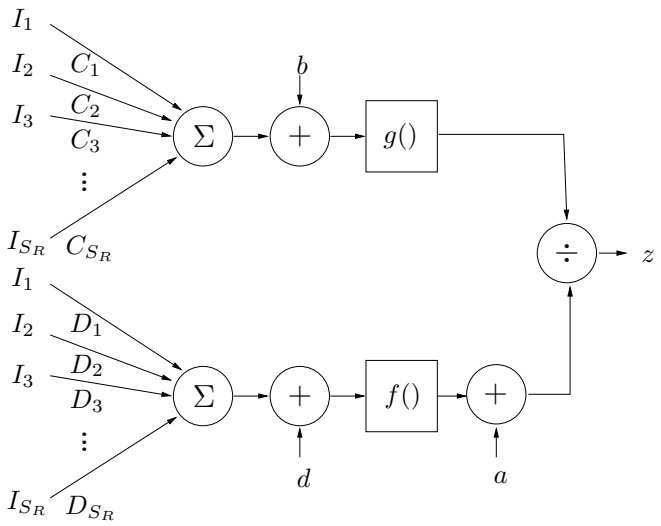


Figure 3.12: The shunting inhibitory neuron model

by steps of two pixels in each direction. In this way, the dimension of the resulting map is about two times smaller, and convolution and sub-sampling layers are combined into one layer. Note that here the sub-sampling does not imply an averaging operation (with trainable parameters) like in the CNN model of LeCun. The bias terms b and d can either be common for each feature map or one can have a different bias for each position of the respective feature map, *i.e.* each shunting inhibitory neuron.

The output layer is rather different from the preceding feature extraction layers. It performs an averaging over 2×2 non-overlapping sub-regions of the preceding convolution map. The model of the output neurons is the Perceptron (*c.f.* section 2.8.2). Thus, the activation of a given neuron is the result of the activation function applied to the weighted sum of the inputs and the bias. The output neurons are fully connected to the preceding feature maps.

Figure 3.13 shows the architecture of one of the first SICoNNets proposed by Tivive and Bouzerdoum [238] and applied to face detection. It has one input layer of size 20×20, two hidden convolution layers with receptive fields of size 5×5 and a neuron output layer. The authors compared different connection schemes for the second convolution layer:

1. *fully connected:* each feature map is connected to every feature map in the first convolution layer

2. *Toeplitz connected:* each feature map of the first convolution layer is connected to several maps in the second convolution layer in an overlapping manner (see Fig. 3.13)

3. *binary connected:* each feature map of the first convolution layer is connected to two feature maps in the second convolution layer. There is no overlapping, *i.e.* each feature map of the second convolution layer is connected to only one map of the preceding layer.

For training a SICoNNet, in principal, any training algorithm for feed-

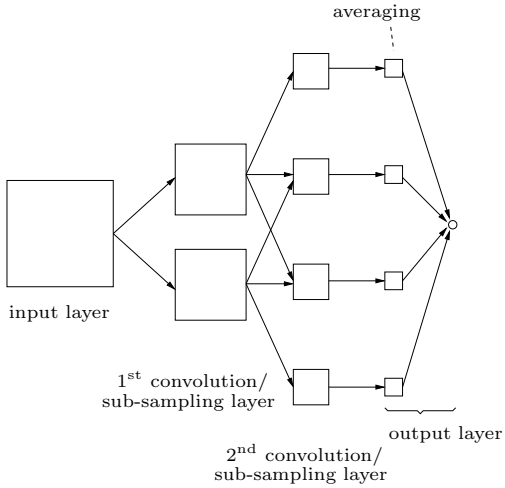


Figure 3.13: The SICoNNet architecture proposed by Tivive *et al.* [238]

forward NNs can be used. The trainable parameters of SICoNNets are the excitatory and inhibitory weights C_j and D_j for each receptive field and the biases b and d as well as the passive decay rate a for each neuron (or each feature map). In [238] the RPROP algorithm (*c.f.* Alg. 5) has been employed for training the face detection SICoNNet. To this end, a classical error function E has been defined:

$$E = \frac{1}{2}(y - t)^2 \ , \tag{3.37}$$

where y is the output of the NN and t is the target output for one particular pattern. Then, the gradient of E w.r.t. each trainable parameter is calculated in order to perform Error Backpropagation as described for MLPs in section 2.8.5.

In a later work Tivive and Bouzerdoom [239] implemented 16 different training methods and compared them in terms of classification rates and convergence speed on a face/non-face classification problem. They also propose two new hybrid training algorithms. The first is based on RPROP, QuickPROP and SuperSAB and is called QRPROP. The second integrates an additional Least Squares (LS) optimization for the final layer and is called QRPROPLS. The performance of these algorithms was tested on three different architectures with different connections schemes: fully connected, Toeplitz connected and binary connected as described above. The experimental results show that Toeplitz connected and binary connected networks slightly outperform fully connected architectures. Good classification rates were obtained with modified Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods, variants of the Levenberg-Marquardt optimization technique and the QRPROPLS method proposed by the authors. In terms of convergence speed, the best results were obtained with QRPROPLS and a hybrid Levenberg-Marquardt/LS algorithm. For details of the training algorithms refer to [239].

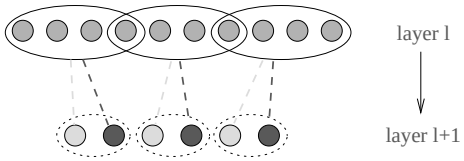


Figure 3.14: The connection scheme of the SCNN proposed by Gepperth. Solid ellipses represent receptive fields. Dotted ellipses represent cells. The dashed lines are receptive field connections where equal line colors stand for shared weights. Here the overlap is one pixel/neuron.

3.4.5 Sparse Convolutional Neural Networks

The idea of *Sparse Convolutional Neural Networks* (SCNN) is to reduce as much as possible the *redundancy* in the feature extraction layers. That means different feature extractors (*e.g.* convolution filters) should be as "independent" as possible and the information they extract should be non-redundant, *i.e.* sparse, and in a way complementary.

Gepperth [85] proposed a SCNN which achieves sparsity with a special type of connectivity and an orthogonalization of the convolution masks. The architecture he presented does not use sub-sampling layers, but a sub-sampling effect is obtained by reducing the overlap of the receptive fields in the convolution layers. Figure 3.4.5 illustrates the connection scheme for the one-dimensional case. Here, instead of putting the result of each convolution into a separate feature map (as in LeCun's model) the outputs of all filters (*i.e.* the weighted sum with the convolution masks) at one particular image location are grouped into a so-called *cell* leading to only one feature map in each layer containing a certain number of cells, one for each receptive field location in the preceding layer. A sigmoidal activation function $\phi(x) = \frac{x}{1+|x|}$ and a trainable bias has been used for each neuron.

In order to reduce the redundancy in the convolution filters, the filter parameters in each cell are orthogonalized after every training epoch. To this end, a matrix $\mathbf{W}$ is formed where each column represents the parameters (*i.e.* weights) of one particular convolution mask. The orthogonalization of $\mathbf{W}$ is then performed in an iterative manner (see [107] for details):

$$\mathbf{W}_0 = \mathbf{W}/||\mathbf{W}|| \tag{3.38}$$

$$\mathbf{W}_{t+1} = 1.5\mathbf{W}_t - 0.5\mathbf{W}_t\mathbf{W}_t^T\mathbf{W}_t \quad . \tag{3.39}$$

In the work presented by Gepperth [85], the error function to minimize is the classical mean squared error of the network outputs w.r.t. the desired outputs. The author experimented with different numbers of layers and varying overlaps. The final layer contained a single neuron and the SCNN has been trained for a visual classification task.

Ranzato *et al.* [193] also proposed a CNN-based approach which learns in an unsupervised manner sparse and also shift-invariant feature extractors. The authors proposed an encoder-decoder framework and applied it to object recognition. The encoder uses a specific type of CNN with alternating convolution and max-pooling layers where each pair of layers is trained separately. A max-pooling layer is a sub-sampling layer where each map is connected to a convolu-

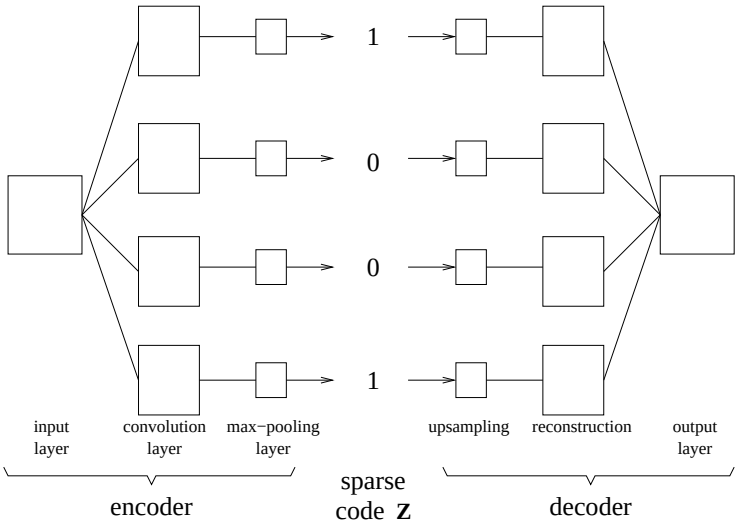


Figure 3.15: The sparse, shift-invariant CNN model proposed by Ranzato *et al.*

tion map in the preceding layer, and each unit in a max-pooling map computes the maximum value within a small neighborhood in the respective convolution map. Figure 3.15 illustrates this architecture with only one convolution and max-pooling layer. To achieve shift-invariance, the transformation parameters $\mathbf{U}$ are retained at the output of the encoder and used to reconstruct the input image in the decoder from the shift-invariant representation $\mathbf{Z}$.

Learning was performed in an EM-like fashion alternating the minimization of the optimal code $\mathbf{Z}^*$ w.r.t. a global energy function and the minimization of the weight parameters of the encoder and decoder, $\mathbf{W_C}$ and $\mathbf{W_D}$, w.r.t. the desired output image $\mathbf{Y}$ and the optimal code $\mathbf{Z}^*$ respectively (see [193] for details).

In order to have the encoder produce *sparse* codes, an additional non-linear logistic function has been used between the encoder and the decoder. This function transforms $z_i(k)$, *i.e.* the i-th component of the k-th training example into a sparse code $\bar{z}_i(k)$:

$$\bar{z}_i(k) = \frac{e^{\beta z_i(k)}}{\zeta_i(k)} \ , \quad \text{with} \quad \zeta_i(k) = e^{\beta z_i(k)} + \frac{1-\eta}{\eta}\,\zeta_i(k-1) \quad , \tag{3.40}$$

where $\eta \in [0,1]$ controls the sparseness of the code and $\beta > 0$ determines the gain of the logistic function. In a way, this function memorizes the history of the activation of a unit, and it can produce a large value close to 1 only if the unit has undergone a long enough quiescent period.

Ranzato *et al.* [193] also extended the simple encoder to a hierarchical two-level feature extractor by using an additional convolution and a max-pooling layer trained separately on the sparse features extracted at the first level. In this way, more complex features can be extracted in an unsupervised way.

3.5 Some Applications

CNNs have been employed in numerous visual pattern recognition systems from different contexts. To give a short overview, we will list here some of the application published in the literature.

As mentioned in the beginning of this chapter, Fukushima was the first to apply a CNN, the so-called Neocognitron, to the problem of handwritten digit recognition [74] and later to handwritten isolated character recognition [78]. A more complex CNN architecture has then been used by Fukushima and Wake [77] to segment and recognize handwritten *connected* characters.

LeCun *et al.* [134] also proposed a CNN for handwritten digit recognition, the so-called LeNet-1 (see section 3.2.2). A later version, the LeNet-5 (see section 3.4.1) [136], has been designed to also recognize letters and special characters, and the authors further propose an integration into a graph framework recognizing the handwritten amounts on bank checks. Neubauer [170] applied a Neocognitron and a modified version to the problem of handwritten digit recognition and compared the results with those of an MLP a SOM and an auto-associative NN. He also conducted experiments on a face recognition task. Recently, Saidane and Garcia [211] applied CNNs to the problem of scene text recognition where the characters to recognize usually show considerable variations in shape, color and background. In another work [212] they further propose a robust method for text binarization employing a specific CNN architecture that is trained to directly output the binarized image of the raw input text image.

Face analysis is another problem that many researchers tackled with CNN-based approaches. Lawrence *et al.* [132] for example use a SOM for dimensionality reduction and then apply a CNN for classification. Further, Fasel [65, 64] uses a CNN architecture with receptive fields of different sizes and applied it to face and facial expression recognition and a combination of both. Osadchy *et al.* [175] presented a combined face detector and pose estimator based on a CNN architecture. And Matsugu *et al.* [157, 158] presented a CNN variant with a spiking neuron model and apply it to face detection and facial expression recognition. The face analysis systems presented by Tivive and Bouzerdoum use shunting inhibitory CNNs (SICoNNets). They applied different architectures to face detection [238] and also eye detection [240]. Finally, Chopra *et al.* [39] proposed a siamese CNN for a face verification problem (see section 3.4.3).

Many other works used CNNs for visual object detection applications in very different contexts. For example, text detection [142], logo detection in TV programs [55], hand detection and tracking [173], lung nodule detection in radiographs [147] or crack detection in sewer pipes [178].

Moreover, LeCun *et al.* [135] introduced a CNN-based system for view-invariant object recognition, *e.g.* planes or cars, on cluttered background. Finally, an original application of CNNs has been presented by LeCun *et al.* [137], namely a camera-based steering system for an autonomous mobile robot avoiding obstacles in an off-road environment.

The large number of applications in which CNNs have been successfully employed illustrates the flexibility and performance of this approach especially in visual pattern recognition problems.

3.6 Conclusion

In this chapter, we presented a feature extraction and classification technique, called Convolutional Neural Network, which is based on a specific type of Neural Network, the Neocognitron, introduced by Fukushima [74]. The characteristics of CNNs are *receptive fields* that extract local features and the concept of "simple" and "complex" cells which, in a hierarchical layer-wise connection scheme, can be used to construct complex, non-linear feature extractors. Further, the principle of weight sharing drastically reduces the number of free parameters in the CNN and therefore improves convergence, reduces computation time and increases its generalization capacity. The use of sub-sampling can further improve the performance of CNNs making the feature extractors invariant to small shifts and distortions.

An important breakthrough concerning the technique of CNNs has been achieved when LeCun *et al.* [134] presented their CNN architecture and trained it with the *Backpropagation* algorithm. Following this approach, the features to extract are not chosen "manually" but learned automatically by the algorithm minimizing a given global error function. A further advantage with this model is that the CNN has not to be trained layer-by-layer but all the parameters are updated at each training iteration allowing an *online* application, which can be important when not all of the training data is available at the beginning.

We also presented some extensions and variants of CNNs and briefly describe how they are trained and applied in different contexts. In a trade-off between simplicity and performance, we chose to adopt and modify the CNN model of LeCun *et al.* [134] for all the face analysis applications in this work. Together with the online Backpropagation training algorithm, it represents a very powerful machine learning technique being able to learn very complex non-linear feature extractors and classifier which can be applied to almost any visual detection and recognition problem. Moreover, the automatic construction of feature extractors and the simultaneous learning of a classifier are clearly an advantage of this technique when applied to face analysis tasks. Obviously, it is rather difficult to "manually" determine which visual features are important in face images as the faces' appearance varies considerably under different conditions.

In the following chapter, we will focus on the problem of face detection in images and different normalization methods in terms of illumination, pose and global alignment. The localization and normalization steps are crucial for further processing of face images, notably for face recognition because the performance of state-of-the-art face recognition systems depends heavily on the precision of these preceding steps.

Chapter 4

Face detection and normalization

4.1 Introduction

Face detection or *face localization* denotes the problem of finding one or several sub-regions in images, each subregion containing a face, *i.e.* determining the position and scale of the respective face regions and sometimes also their rotation angle in the image plane. Face detection algorithms mostly represent their results as bounding boxes covering the regions of the image which are supposed to contain faces, *i.e.* one bounding box per face. Note that most of the face analysis tasks, *e.g.* face recognition, facial expression recognition, facial feature detection, require the localization of the face as an initial step.

Face detection can imply a pre-processing step conditioning the input image and performing a kind of normalization, *e.g.* converting the image to gray-scale, equalizing the intensity histogram of the image or image regions. Usually, this pre-processing is considered to be part of the face detection algorithm.

Additionally, there can be a post-processing step further normalizing the result of the face detection procedure, *e.g.* the bounding boxes. Possible operations are illumination normalization, pose normalization or estimation, or face alignment, that means the global alignment of the bounding boxes w.r.t. the face images they cover. We will explain these procedures in more detail in sections 4.3-4.5, and in the following we will refer to them as *face normalization*. Face normalization is an important step in many face analysis applications because it eliminates visual information irrelevant to the respective task. For example, in face recognition, varying illumination conditions can cause face images of the same person to appear considerably different while the face images of two different persons under the same illumination conditions might look very similar. Thus, without a preceding illumination normalization the overall classification might be rather difficult.

In the following sections, we will first outline the state-of-the-art in face detection and describe in more detail one method based on CNNs that has been used throughout this work, the Convolutional Face Finder [81]. Then, we will present a brief overview of current illumination normalization and pose estimation techniques, and finally we will focus on the process of face alignment

and present a novel system for global affine alignment of face images using CNNs.

4.2 Face detection

4.2.1 Introduction

In this section we will first give an overview of current face detection techniques presented in the literature and then describe one approach in more detail: the Convolutional Face Finder proposed by Garcia and Delakis [81]. This system is based on CNNs and outperforms other state-on-the-art face detection methods. Face detection algorithms are mostly compared in terms of their detection rate (*i.e.* number of correctly detected faces) vs. their false alarm rate (*i.e.* number of non-face regions erroneously detected as faces) measured on public face databases, for example the CMU face database [206]. Clearly, one seeks a method that maximizes the detection rate while minimizing the false alarm rate. Usually, the latter is a rather difficult task as the subset of non-face images of a given size is considerably larger and more complex than the subset of images representing faces.

4.2.2 State-of-the-art

Many approaches to face detection have been proposed in the literature. Hjelmås *et al.* [98] and Yang *et al.* [266] presented surveys on the most important of them. They can be roughly divided into two categories: *template-based* and *feature-based* methods. In template-based approaches, the input image is scanned at each possible location by extracting a subregion of varying scale from it and classifying it as face or non-face.

Sung and Poggio [231] presented such an approach using Neural Networks. They first modeled the distribution of face and non-face images (of size 19×19 pixels) using a clustering algorithm and then calculate for every training example the distance to each cluster center forming 24 element feature vectors. Then, they trained a Neural Network with these feature vectors to distinguish between faces and non-faces. Osuna *et al.* [176] proposed a similar approach but using Support Vector Machines (SVM) instead of a Neural Network.

Colmenarez and Huang [40] proposed a system based on Kullback relative information (Kullback divergence) to measure the difference between joint-histograms computed for each pair of pixels in the face images of the training set, for the classes of faces and non-faces. With the two trained probability models one is then able to localize faces by scanning the input image at different scales and determining the sub-regions where a pre-defined likelihood measure is above a certain threshold.

Rowley *et al.* [207] used a Neural Network architecture with a specific connection of the hidden layer and applied it directly on the pixels of 20×20 sub-windows to be classified as face or non-face. In order to reduce the number of false alarms they trained several Neural Networks and used an arbitration strategy for the final classification. Further, they accelerated their face detection system by a two-stage approach using a simple Neural Network to filter

out possible face candidates in the first stage and refine the classification in the second stage.

Roth *et al.* [202] proposed a face detector based on a learning architecture called SNoW (Sparse Network of Winnows), which consists of two linear threshold units, representing the classes of faces and non-faces, that operate on an input space of Boolean features. Features like intensity mean, intensity and variance were first extracted from a series of sub-windows from the face window and then discretized into a predefined number of classes to give boolean features in a 135,424-dimensional feature space. The system was trained with a simple learning rule, which increases and decreases weights in cases of misclassification, in order to classify face and non-face boolean features.

Schneiderman and Kanade [218] presented a face detection method based on a locally sampled three-level wavelet decomposition. Several sets of wavelet coefficients were extracted from chosen sub-bands of the wavelet tree. The coefficients were re-quantized to three levels and probabilistic density functions were built using histograms. Then, they applied Bayes' rule for the classification between face and non-face patterns.

Heisele *et al.* [97] proposed an approach based on a Support Vector Machine. They trained the SVM to classify the whole input pattern as face or non-face using a set of training images of size 19×19 pixels. The features they extracted are the histogram-normalized image, gradients computed by a Sobel filter and the outputs of the convolution with some specific Haar wavelets filters. Then, they compared this method with a feature detection-based approach (see section 5.2).

Fröba and Küllbeck [72] described a detection algorithm calculating orientation maps of the input image at different scales. The resulting maps were scanned using an orientation map template of the face.

An efficient method called "Boosted Cascade Detector" has been proposed by Viola and Jones [250]. They made use of simple classifiers summing pixel values from adjacent regions of the candidate image patch. Then they combined many of these weak classifiers and weighted them in a certain way to finally obtain a strong classifier being able to distinguish the patch between face or non-face. The selection and weighting of these weak classifiers was performed by the "Adaboost" algorithm, explained in more detail in section 2.5.

S. Li *et al.* [143] extended the idea of Viola and Jones and proposed a multi-view face detection method. They applied different levels of detectors, more or less specialized to specific poses. In a coarse-to-fine approach, the range of detectable poses is more and more restricted. If all detectors at a particular level fail the candidate region is rejected.

Other techniques are based on standard multivariate statistical analysis. Yang *et al.* [265], for example, presented two methods which seek to represent the manifold of human faces as a set of subclasses. In the first method, a mixture of factor analyzers was used to perform clustering and local dimensionality reduction within each obtained cluster. The second method uses Kohonen's self-organizing maps for clustering, Fisher's linear discriminant to find an optimal projection for pattern classification and a Gaussian distribution to model the class-conditional density function of the projected samples for each class. Maximum likelihood estimates were used for the parameters of the class-conditional density functions and the decision rule.

Féraud *et al.* [67] proposed a Neural Network approach, based on constrained

generative models (CGM), which are auto-associative fully connected MLPs with three large layers of weights, trained to perform a nonlinear dimensionality reduction similar to PCA. Classification is obtained by considering the reconstruction errors of the CGMs. The best results were reported using a combination of CGMs via a conditional mixture and an MLP filtering out unlikely candidates. As the computational cost of this method is high, some pre-filtering operations are required, such as skin color and motion segmentation. Like in the previous neural based approaches, every tested sub-window was preprocessed using the approach of Sung and Poggio [231].

More recently, Garcia and Delakis [81] presented a face detection method using Convolutional Neural Networks (CNN). Here, specialized filters and classifiers were automatically and conjointly learned by the Neural Network using face and non-face examples and a specific bootstrapping algorithm. Being particularly robust w.r.t. noise and partial occlusions and producing very few false alarms, this technique, so far, shows the best performance on difficult public face databases such as the CMU database [206]. In section 4.2.3 we will describe this method in more detail.

Finally, Osadchy *et al.* [175] also presented a face detection system based on CNNs. They employed an network architecture similar to LeNet-5 (see section 3.4.1) and trained it to map face images with known head pose onto points in a low-dimensional manifold parameterized by pose and non-face image onto points far away from that manifold. Once the CNN has been trained, it can classify image sub-regions as face or non-face by applying the mapping to them and calculating the distance between the projected vector and the analytical manifold. The advantage of this method is that it performs an implicit pose estimation at the same time as face localization. Additionally, an synergistic effect of the integration of both has been shown. Another advantage is the relatively large range of pose handled by the system, *i.e.* $[-90, +90]$ degrees of left/right rotation (yaw) and [-45,+45] degrees of up/down rotation (roll). However, the performance of the method in terms of detection rate and false alarm rate measured on the CMU data set is inferior to the system proposed by Garcia *et al.* [81].

Table 4.1 summarizes the performance in terms of detection rate of some of the previously mentioned methods. It can be observed that the method by Garcia and Delakis compares favorably with the others, especially for low numbers of false alarms. This shows that this approach separates the face and non-face spaces in a robust and balanced way. For larger number of false alarms, its results are equivalent to the ones reported for the other methods. It suggests that all these detectors reach very similar maximal detection limits.

The previously listed methods are all template-based approaches as they try to match a global face model onto sub-regions of the image to process. *Feature based* approaches [141, 42, 268, 116, 83, 94], however, search for particular facial features in the input image and, most often, reduce the number of candidates by probabilistic models of feature constellations. The detection of single features can make these methods more robust to out-of-plane rotation, or partial occlusions of the face. However, in contrast to template-based methods, they are unsuitable for the detection of low-resolution faces, *e.g.* 20×20 pixels. In chapter 5, we will outline some facial feature detection methods that have been employed for face detection.

Unfortunately, most of the feature-based face detection approaches proposed

Table 4.1: Detection rate vs. false alarm rate of selected face detection methods on the CMU test set.

	False alarms				
Face detector	0	10	31	65	167
Rowley *et al.* [207]	-	83.2%	86.0%	-	90.1%
Schneiderman *et al.* [218]	-	-	-	94.4%	
Li *et al.* [143]	-	83.6%	90.2%	-	-
Viola and Jones [250]	-	76.1%	88.4%	92.0%	93.9%
Osadchy *et al.* [175]	-	-	-	83.0%	88.0%
Garcia *et al.* [81]	88.8%	90.5%	91.5%	92.3%	93.1%

in the literature use different evaluation methods and/or databases which makes their comparison in terms of performance and precision impossible.

In the following section, we will describe in more detail the Convolutional Face Finder (CFF) presented by Garcia and Delakis [81] because it achieves good performance in terms of detection rate and because it is based on Convolutional Neural Networks.

4.2.3 Convolutional Face Finder

Introduction

The CNN-based face detection system proposed by Garcia and Delakis [81] and dubbed *Convolutional Face Finder* (CFF) can be classified as a template matching technique as it verifies the presence or absence of a face at each possible location in the face image employing a sliding-window approach. The CFF has been proven to be one of the best performing face detection methods up to the present regarding its high detection rate and low false alarm rate on difficult real-world images. Therefore, we will use this system throughout the rest of this work and further improve it by an additional face alignment system explained in section 4.5.

Architecture

The architecture of the CFF is essentially a CNN with 7 layers including the input layer. It is based on the CNN model of LeCun *et al.* [134] but has a different number of feature maps and neurons and notably a different connection scheme between the layers. Figure 4.1 shows the basic architecture of the CFF. There are two alternating convolutional and sub-sampling layers C_1, S_1 and C_2, S_2 followed by two neuron layers N_1 and N_2. The size of the input layer, *i.e.* the retina, is 32×36 pixels.

Layer C_1 is composed of four feature maps of size 28×32 pixels. Each unit in each feature map is connected to a 5×5 neighborhood into the input retina of size 32×36. Each feature map unit computes a weighted sum of its input by 25 (5×5) trainable coefficients, i.e. the convolution kernel, and adds a trainable

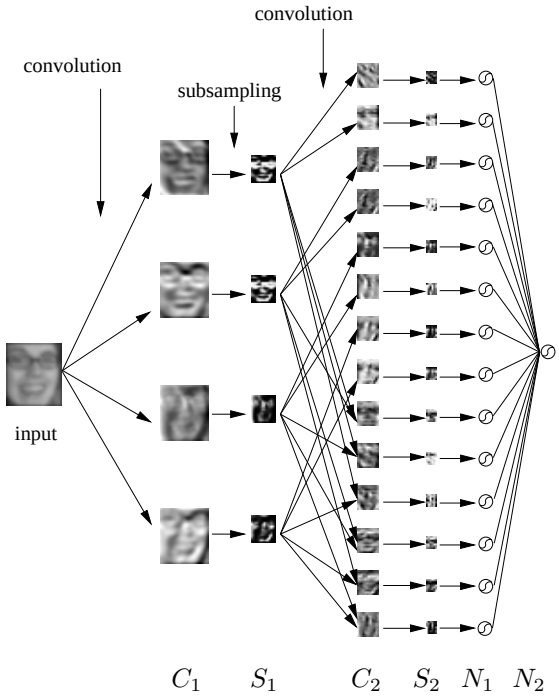


Figure 4.1: The architecture of the Convolutional Face Finder

bias.

Layer S_1 is composed of four feature maps of size 14×16 pixels, each connected to one feature map in C_1. The receptive field of each unit is a 2×2 area in the previous layer's corresponding feature map. Each unit computes the average of its four inputs, multiplies it by a trainable coefficient, adds a trainable bias, and the result passes through a hyperbolic tangent function, *i.e.* the activation function of the unit.

Layer C_2 contains 14 feature maps performing 3×3 convolutions. Here, outputs of different feature maps are fused in order to help in combining different features, thus in extracting more complex information. Each of the four subsampled feature maps of S_1 provides inputs to two different feature maps of C_2. This results in the first eight feature maps of C_2. Each of the other six feature maps of C_2 takes its input from one of the possible pairs of different feature maps of S_1. Table 4.2 illustrated the connection scheme of layer C_2. Consequently, layer C_2 has 14 feature maps of size 12×14.

Layers N_1 and N_2 contain classical neural units. These layers act as a classifier, whereas the previous ones act as feature extractors. In layer N_1, each of the 14 neurons is fully connected to all units of only one corresponding feature map of S_2. The single neuron of layer N_2 is fully connected to all neurons of layer N_1. The units in layers N_1 and N_2 perform the classical dot product between their input vector and their weight vector to which a bias is added. A subsequent application of the hyperbolic tangent function produces an output between -1.0 and $+1.0$. The output of neuron N_2 is used to classify the input image as a *non-face*, if its value is negative, or as a *face*, if its value is positive.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	X	X							X	X	X			
1			X	X					X			X	X	
2					X	X				X		X		X
3							X	X			X		X	X

Table 4.2: The connection scheme of layer $C2$ of the Convolutional Face Finder: a cross indicates a connection between a particular feature map in $S1$ (row) and a convolution map in $C2$ (column)

Figure 4.2: Some patterns used for training. The first three rows show some highly variable face patterns. The last row contains some examples of non-face patterns produced by the bootstrapping procedure (Courtesy of C. Garcia).

Training

In order to train the CNN, the authors used a set of about 3,700 manually cropped, highly variable face images extracted from various sources of the Internet and from scanned newspapers. The collected images are chosen to effectively capture the variability and the richness of natural data in order to train the system for operating in uncontrolled environments. No intensity normalization such as histogram equalization is performed on the cropped faces. In order to create more examples and to enhance the tolerance to small in-plane rotations and variations in intensity, a series of transformations are applied to the initial set of face examples, e.g. small rotations, mirroring and smoothing. The set of initial negative (i.e. non-face) examples is built by randomly cropping regions out of images that do not contain any faces. Then, this set is gradually augmented by a so-called bootstrapping procedure which, every 60 training iterations, applies the Neural Network on scenery images not containing any face and collects the false alarms that give a response above a certain threshold. This threshold is decreased at each step. Thus, the frontier between positive and negative examples is gradually refined. Figure 4.2 shows some example images of faces and non-faces used for training. The training algorithm is the standard online Backpropagation with momentum which was slightly adapted to cope with weight sharing (*c.f.* Alg. 8 on page 57). For further details on the training procedure refer to [81].

The face localization procedure

Figure 4.3 depicts the different steps of face localization with the CFF in a gray-scale image containing three faces. In order to detect faces of different

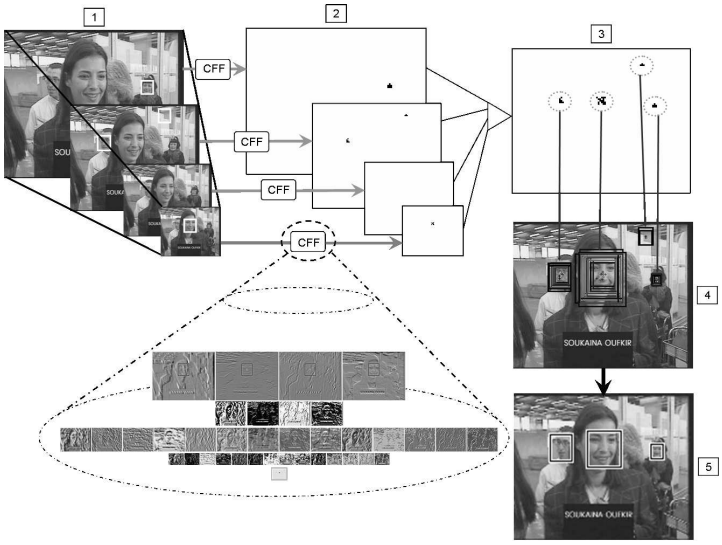


Figure 4.3: The face localization procedure. (1) creation of a multi-scale pyramid from the original image. (2) convolution of each image of the pyramid by the CFF. (3) projection of face candidates to the original scale and fusion of overlapping face candidates. (4) application of the CFF in a fine pyramid centered at each face candidate position. (5) classification of each face candidate according to the volume of positive answers in the corresponding fine pyramid (Courtesy of C. Garcia).

sizes, the input image is repeatedly sub-sampled by a factor of 1.2, resulting in a pyramid of images (step 1). Each image of the pyramid is then entirely filtered by the CFF (step 2). In fact, this corresponds to a Space Displacement Neural Network (SDNN) applied at several scales (see section 3.4.2). For each image of the pyramid an image containing the network results is obtained. Because of the successive convolutions and sub-sampling operations, this image is approximately four times smaller than the original one. This fast procedure may be seen as corresponding to the application of the network retina at every location of the input image with a step of four pixels in both axis directions, without computational redundancy.

After processing by this detection pipeline, face candidates (pixels with positive values in the result image) in each scale are mapped back to the input image scale (step 3). They are then grouped according to their proximity in image and scale spaces. Each group of face candidates is fused in a representative face whose center and size are computed as the centroids of the centers and sizes of the grouped faces, weighted by their individual network responses. After applying this grouping algorithm, the set of remaining representative face candidates serve as a basis for the next stage of the algorithm performing *fine* face localization and, eventually, false alarm dismissal.

That means, a local search procedure is performed in an area around each face candidate center in image scale-space (step 4). A reduced search space centered at the face candidate position is defined in image scale-space for precise localization of the face candidate. It corresponds to a small pyramid centered

at the face candidate center position covering ten equally distant scales varying from 0.8 to 1.5 times the scale of the face candidate. For every scale, the presence of a face is evaluated on a rescaled grid of 16x16 pixels around the corresponding face candidate center position. In order to discriminate true faces from false alarms, the authors took into account both number and values of positive answers. Therefore, they considered the *volume* of positive answers (the sum of positive answer values) in the local pyramid in order to take the classification decision. Based on the experiments described in the following section, a face candidate is classified as face if its corresponding volume is greater than a given threshold $ThrVol$ (step 5). The bottom-right image of Fig. 4.3 shows the positions and sizes of the faces detected after local search. One can notice that the false alarm (up right in the image), previously detected in step 4, with a low volume after local search, has been removed using the volume threshold criterion.

Experimental results

The authors evaluated their method on three different test sets: the first one is the *CMU* test set [207] consisting of 130 images and containing 23 images of the *MIT* test set [231]. *CMU-125* and *MIT-20* are subsets of CMU and MIT respectively, excluding hand-drawn and cartoon faces. The second set is called *Web* test set and contains 215 images randomly chosen from submissions to an interactive online demonstration of the face detection system. Finally, the third test set is referred to as *Cinema* and consists of 162 images extracted from various movies and showing faces under very difficult conditions.

Only a few papers, including [265], address the definition of *what is a correctly detected face* [98]. In the experiments in [81], a detected face is considered valid if the face window is not 20% bigger that the real face area and contains both eyes and mouth. In some papers, reported results of selected methods on the face databases are difficult to interpret. For a given approach, some results correspond to the maximal detection rate with a high number of false alarms and others to a lower detection rate with a smaller number of false alarms. Indeed, most face detectors can adjust parameters (usually a threshold) influencing detection and false alarm rates. The detection rate is the ratio between the number of successful detections and the number of labeled faces in the test set. The false alarm rate is the ratio between the number of false positive detection and the number of scanned windows. This can be illustrated in terms of a Receiver Operator Characteristic (ROC) curve to show the detection rate versus the false alarm rate for various values of the threshold. Figure 4.4 depicts the ROC curves for the *CMU*, the *Web* and the *Cinema* test sets. Each point on a curve corresponds to a given threshold $ThrVol$ applied to the volume of positive answers in order to classify an image area into face or non-face. Note that quite high detection rates were obtained with zero false alarms, *i.e.* 88.8%, 90.5% and 80.4% for the *CMU*, the *Web* and the *Cinema* test sets respectively. On the other hand, the maximum detection rates are 93.3% with 197 false alarms, 98.0% with 108 false alarms and 95.3% with 104 false alarms, for the *CMU*, the *Web* and the *Cinema* test sets respectively. These results are summarized in Table 4.1 on page 75.

Table 4.3 lists the detection rates for various numbers of false detections for the CFF (with $ThrVol = 17.0$) as well as for other published systems as

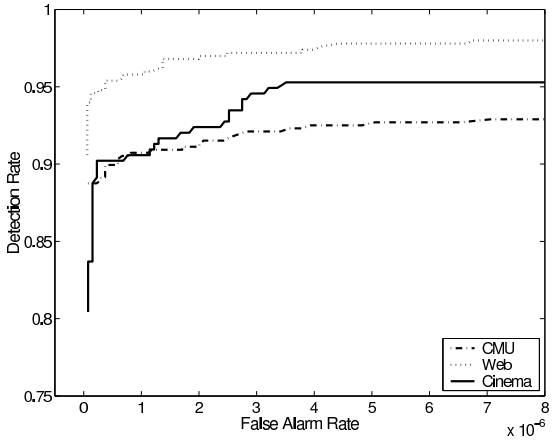


Figure 4.4: The ROC curves obtained for the *CMU*, the *Web* and the *Cinema* test sets. Every point on a curve correspond to a given threshold value $ThrVol$ of the volume of positive answers (Courtesy of C. Garcia).

reported in [250, 143]. It can be observed that the CFF method compares favorably with the others, especially for low numbers of false alarms. This shows that the approach separates face and non-face space in a robust and balanced way. For larger number of false alarms, the results of the compared methods are equivalent.

Finally, Fig. 4.5 shows some examples of the detection results for the CMU test set.

Implementation on embedded platforms

Roux, Mamalet and Garcia [203] successfully implemented the CFF system on various embedded platforms like ARM (Xscale, IMX21), DSP (Starcore) and orange SPVM 3000. To this end, extensive memory and computational optimizations have been conducted on the original implementation. For example, all parameters and calculations have been transformed from floating point to fixed point arithmetic without any loss of precision and effectiveness of the overall face detection.

Further optimizations exploiting the parallel computation of embedded processors have been performed. For example, consider the convolution of the input image by a mask of 5×5 coefficients. Instead of processing the image at each location of the receptive field and loading the 25 coefficients into memory at each time, the whole image is scanned 25 times with a different coefficient and four load/store operations are performed in parallel leading to an overall gain of these operations of about 25%.

Also, the amount of used memory has been considerably reduced by conserving intermediate results in the layers C_1 to S_2 when computing the activation of adjacent neurons in layer N_1.

Finally, an overall speed-up factor of 55 compared to the original implementation has been achieved on the Xscale platform using these optimization techniques. Table 4.4 shows the processing speed on different platforms.

Face Detector	CMU	CMU-125	MIT	MIT-20
Colmenarez *et al.* [40]	93.9%/8122			
Féraud *et al.* [67]	86.0%/8			
Yang *et al.* [265]		93.6%/74		91.5%/1
Osuna *et al.* [176]			74.2%/20	
Roth *et al.* [202]		94.8%/78		94.1%/3
Rowley *et al.* [207]	86.2%/23		84.5%/8	
Schneiderman *et al.* [218]		94.4%/65		
Sung *et al.* [231]			79.9%/5	
Viola *et al.* [250, 251]	88.4%/31		77.8%/5	
Li *et al.* [143]	90.2%/31			
Osadchy *et al.* [176]	83.0%/65			
CFF [81]	90.3%/8	90.5%/8	90.1%/7	90.2%/5

Table 4.3: Face detection results in terms of percentage of good detection / number of false alarms, for the CMU and MIT test sets

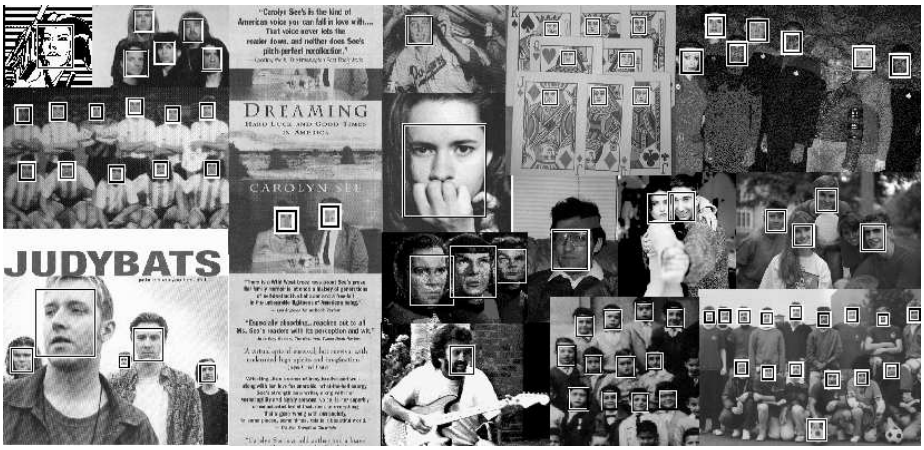


Figure 4.5: Some face detection results of the CFF obtained with the CMU test set (Courtesy of C. Garcia).

	Xscale PXA27x @ 624MHz	Starcore SC140 @ 275MHz	Pentium IV @ 3.2GHz
Floating point version	0.3 fps	-	10 fps
Optimized version	16.5 fps	35 fps	180 fps

Table 4.4: Execution speed of the CFF on different platforms (in frames per second) on QCIF images (176×144 pixels)

Conclusion

In this section, we described a face detection method based on a CNN, the Convolutional Face Finder of Garcia and Delakis [81], and showed the effectiveness and efficiency of the overall system compared to state-of-the-art face detection algorithms. Since the results in terms of detection rate, false alarm rate and execution time are superior to other existing methods, the CFF system will be used throughout the rest of this work.

We will now describe different post-processing techniques once the image region containing a given face is determined. In general this type of normalization is supposed to facilitate further facial analysis, *e.g.* face recognition, and is thus considered an important step in many face image-related applications. *Illumination normalization* and *pose estimation* are two of the most common such operations that can be performed after a face has been detected. We will briefly outline some existing approaches to these problems in the following.

4.3 Illumination Normalization

Illumination normalization is concerned with the problem of transforming a face image taken under some lighting condition into an image having a different ("normal") illumination without compromising the appearance of facial characteristics and thus the ability to detect facial features and/or to recognize the face. A normal illumination means, for example, a light source in front of the face or some ambient lighting. An inherent problem is the removal of shadows that the face casts on itself. This is a very complicated task as the appearance of faces under varying illumination follows a highly non-linear function. Note that here, we are only considering the normalization of *luminance* but not *color* since most of the face analysis methods operate on gray-scale images.

There exist also some techniques that can't actually be considered as *normalizing* illumination but rather make the input image less sensitive to illumination changes.

For example, one straightforward approach based on physiological evidence is to use a *logarithmic transformation* of the image intensities. However, the effectiveness of this technique depends on the face analysis approach utilized and thus has to be determined experimentally.

Another approach that many solutions adopt is to use *first or second order derivatives* of the input image in one or several directions. Edge maps are a classic example of this type of approach. This achieves a certain invariance to changes in the ambient light but not to changes in the direction of the light source as it is often the case in natural images.

One of the most common illumination normalization methods used in many of the face analysis methods proposed in this work is *histogram equalization*. Here, the histogram of pixel intensity values of an image is mapped onto the full range of gray values. Albeit very straightforward to calculate, this transformation only compensates for variations of the overall brightness of the image and cannot cope with illumination from point light sources of different directions as well as shadows.

An approach specific to faces is the *mapping of a linear function* onto the face image [207]. Here, an oval mask is put on the image containing the face in order

to ignore background pixels, a linear function is then fit to the intensity values of the image and subtracted out. Finally, histogram equalization is applied. This technique copes with some extreme lighting conditions, notably from the side.

Shashua *et al.* [225] proposed an illumination invariant image representation called the *quotient image*. Using a database, called the bootstrap set, containing several faces, each under three different lighting condition, they present an analytical method to calculate the quotient image of any novel face under any illumination.

Belhumeur and Kriegman [15] showed that the set of images of an object in a fixed pose under varying illumination forms a convex cone in the image space. Georghiades *et al.* [84] model this space by low-dimensional linear subspaces. Basri and Jacobs [13] show that it can be approximated by a 9-dimensional linear subspace.

A well-known approach introduced by Horn *et al.* [99, 110, 100] is called Shape From Shading (SFS) where faces are considered to be Lambertian surfaces and the surface normals and/or albedos are estimated by using several face images taken from the same viewpoint under different illumination. Having estimated the shape of the face, it can then be re-rendered under any illumination. However, this is an ill-posed problem and most of the existing works overcome this by imposing certain constraints. For example, Zhou *et al.* [273] imply rank and symmetry constraints into the model in order to separate illumination from the observed appearance of a face. The approach proposed by Sim and Kanade [228] incorporates statistical prior knowledge about faces into the model, and it is then able to render a new face under novel illumination using only one example image.

4.4 Pose Estimation

As mentioned in chapter 1, varying head pose is one of the most important sources of variation in face images and makes further processing, like recognition, very difficult because the appearance of a face image under varying pose undergoes a highly non-linear transformation. Thus, the estimation and/or normalization, *i.e.* correction, of pose is a crucial issue in face analysis applications.

Head pose is mostly parameterized by three rotation angles corresponding to a orthogonal coordinate system aligned with the image plane. We define the image plane to be parallel with the x/y-axes, and we will denote the axes and the rotations around these axes as follows:

- x-axis: pitch (corresponds to an up/down movement of the head)
- y-axis: yaw (corresponds to a left/right movement of the head)
- z-axis: roll (corresponds to a tilting head movement)

Figure 4.6 illustrates the defined axes w.r.t. a frontal head. We also distinguish between in-plane rotations (roll) and in-depth rotations (yaw, pitch). The latter can be normalized by rotating the input image such that the face is in an upright position. This type of correction is *linear* and part of what we call *face alignment* in this work and which is described in detail in the following section. In this section, we will outline some pose estimation methods for the more complicated

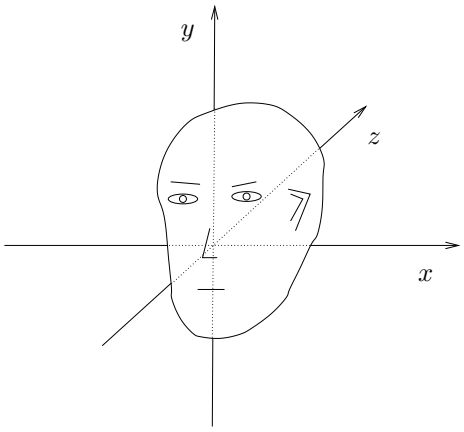


Figure 4.6: The three rotation axes defined with respect to a frontal head

case of in-depth rotation, *i.e.* yaw and pitch. Note that some of these methods require the face image to be well-aligned, *i.e.* centered, in an upright position and in a predefined scale. Further, some of the approaches described in the following only account for yaw, others for both, yaw and pitch. Also, the possible ranges of the parameters differ between the methods, *e.g.* $[-90, +90]$ degrees for yaw. And finally, the precision can vary drastically between the different approaches, *i.e.* some are only trained to distinguish between semi-profiles and frontal face images and others determine the rotation angle with a precision of some degrees. Despite these big differences, we will give an overview on the existing pose estimation methods while focusing on the algorithmic properties rather than the technical details.

Pose estimation can be *integrated* into face detection or recognition methods [181, 231, 171, 218, 143, 104]. In this case, the terms *view-based face detection* or *view-based face recognition* are often used. That means, the total range of poses is divided into a finite number of sub-ranges for each of which a view-specific face detector (or recognizer) is trained. The trained systems are then applied in parallel, and a final decision function determines the classification result and, implicitly, the respective pose.

There are also systems which perform a pose estimation on each possible location of the input image before the actual face detection, *i.e.* before deciding if the processed image patch at a particular location is a face or not. For example, Li *et al.* [144] first applied a Sobel filter on the respective sub-regions, then they performed a PCA to reduce the dimension of the vectors and finally used two SVMs to estimate yaw and pitch of the face candidate. The image patch is then sent to a view-specific face detector which classifies it as face or non-face.

Jones and Viola [118] also performed pose estimation before the actual face detection. They employed Haar-like wavelets and decision trees to determine the pose of a face candidate and then sent the image region to an Adaboost-based face detector similar to [250].

Other methods employ a Shape From Shading (SFS) approach introduced by Horn *et al.* [99, 110, 100] (see section 4.3), *i.e.* they determine a 3D mesh model of the face and then re-synthesize a face image under frontal, normalized

pose. However, unless specific constraints are enforced on the algorithm, several images of the same face under different illumination are needed, which makes this approach rather impractical in common face analysis applications.

A different approach based on *linear object classes* has been proposed by Vetter *et al.* [248]. It is rather a pose *normalization* than a simple pose parameter *estimation* method. The authors assumed that the 3D shape of an object (and 2D projections of 3D objects) of a given view can be represented by a linear combination of prototypical objects (of the same view). It follows that a rotated view of the object is a linear combination of the rotated views of the prototype objects. In this way, one can synthesize rotated views of face images from a single example view. In [21], this algorithm has been used in a multi-view face recognition system to create virtual views from a single input image. Lando and Edelman [129] used a comparable example-based technique for the same purpose.

The above mentioned methods are rather special cases of pose normalization techniques as they are either an integral part of a face detection/recognition system or need several input face images, or they synthesize a face in a given (normalized) pose instead of estimating the underlying pose parameters. However, the methods outlined in the following are pure pose estimation techniques as they process face images and estimate the parameters of the corresponding pose of the face. These approaches can be divided into two main categories: *feature-based* and *template-based* (or appearance-based) methods.

Feature-based methods localize specific facial features and then compare them with some probabilistic or geometric model in order to determine the head pose. The method of Azarbayejani *et al.* [4], for example, detects feature points having a large Hessian w.r.t. image intensity assuming that these points correspond to eye corners, pupils, nostrils etc. The authors applied their algorithm on videos and tracked the detected points using an Extended Kalman Filter (EKF).

Lee *et al.* [139] used a deformable 3D model composed of an edge model, a color region model and a wireframe model in order to synthesize images of faces in arbitrary poses. An iterative energy minimization approach has been employed to match the generic model to a given test face image and thus estimate the corresponding head pose.

Chen *et al.* [38] proposed a method segmenting the two separate regions of the face and the hair respectively and then estimate the x, y and z rotation of the head using geometrical features of the two regions, such as area, center and axis of least inertia.

Approaches based on Bunch Graph Matching (BGM) have been proposed by Krüger *et al.* [127] and Elagin *et al.* [61]. Here, the local appearance of the faces has been described with Gabor jets, *i.e.* set of Gabor wavelet filter responses, which are organized in a two-dimensional graph that is fit to the face image by minimizing a global energy function.

In a later work, Krüger *et al.* [126] presented a pose estimation approach similar to BGMs using Gabor Wavelet Networks (GWN). Unlike the separate representation of the graph and the wavelets in BGMs, GWNs are able to simultaneously encode the local appearance of the face image and geometrical properties of local features. They used a Linear Local Map (LLM) [198], a linear approximation method related to Self-Organizing Maps and Generalized Radial Basis Functions, for a final classification step.

Lanitis *et al.* [131] presented a pose estimation method based on Active Shape Models where a statistical model of the shape of a face using landmark points is iteratively fit to the face image (see section 2.3).

In contrast to feature-based approaches, template-based methods do not localize specific facial features but treat the face image as a whole. Schiele *et al.* [217], for example, presented a method where the face region is segmented from the background and the hair regions by a skin-color-based segmentation algorithm. The pose was then estimated by a MLP that classifies the resulting pattern. MLPs have also been used by Zhao *et al.* [270] and Brown *et al.* [31].

Huang *et al.* [105] use SVMs for left/right/frontal pose classification. In their approach, one SVM for each pose was trained on the raw face images and with two different kernels, a polynomial and a RBF kernel.

Rae *et al.* [192] proposed a method that filters the face images with Gabor wavelets placed on a circular grid on the input image and then classifies the respective head pose by a LLM.

McKenna *et al.* [159] also presented a method based on Gabor Wavelet filters. The head pose was classified by comparing the resulting filtered image with a certain number of template images, each corresponding to a certain yaw and pitch.

Baluja [10] classified five different head poses by different probabilistic models, *e.g.* a Naive-Bayes model, which was built using labeled and unlabeled binarized face images. Another approach, presented by Ba *et al.* [5], uses Gaussian Mixture Models (GMM) to represent the different head poses.

Niyogi and Freeman [172] employed a template matching technique based on a vector quantization method called Tree-Structured Vector Quantizer (TSVQ), *i.e.* the training examples (the templates) together with their respective pose were organized in a tree-like structure using PCA and clustering.

Other methods are based on the concept of Pose Eigenspace (PES) [87] which is a linear sub-space spanned by the first three eigenvectors of the face images' covariance matrix. These three directions are assumed to account for the majority of appearance variations due to changing pose. Further, a face image with varying pose represents a continuous manifold in this sub-space. The method proposed by Motwani [166] builds a PES with wavelet filtered face images and classifies face pose by calculating the Euclidean distance between the projected vectors of given test image and the training images respectively. Wei *et al.* [256] also used a PES-based approach using Gabor wavelet filters and a Mahalanobis distance measure.

Finally, a different approach based on Dynamic Space Warping (DSW) has been proposed by Yegnanarayana *et al.* [267]. Here, a face image under a given pose is horizontally warped to best match a frontal face and the respective warping path is analyzed in order to estimate the yaw pose angle.

4.5 Face Alignment

4.5.1 Introduction

Face alignment or *face registration* designates a normalization procedure where the bounding boxes coming from a face detector are transformed in such a way that the faces are well aligned with them. "*Well aligned*" means most

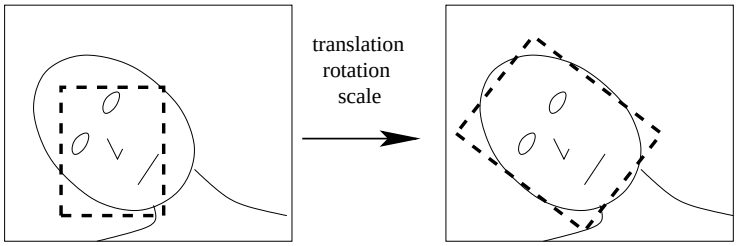


Figure 4.7: The face alignment process

often: centered and in a predefined scale and orientation. Figure 4.7 illustrates this procedure. Thus, the transformation is in most cases *affine* and the face images cropped along the boundaries of the resulting rectangles have specific facial features, such as eyes, nose, mouth, approximately at predefined relative positions in the image.

This step is crucial for further effective face analysis, *e.g.* face recognition. Shan *et al.* [224] and Rentzeperis *et al.* [196] showed that slight mis-alignments, *i.e.* x and y translation, rotation or scale changes, cause a considerable performance drop of most of the current face recognition methods.

There have been different approaches to overcome the mis-alignment problem in face recognition. Shan *et al.* [224], for example, added virtual examples with small transformations to the training set of the face recognition system and, thus, made it more robust. Martinez [153] additionally modeled the distribution in the feature space under varying translation by a mixture of Gaussians. However, these approaches can only cope with relatively small variations.

4.5.2 State-of-the-art

Existing approaches can be divided into two main groups: approaches based on facial feature detection and global matching approaches, most of the published methods belonging to the first group.

Berg *et al.* [20], for example, used a SVM-based feature detector to detect eye and mouth corners and the nose and then applied an affine transformation to align the face images such that eye and mouth centers are at predefined positions. Wiskott *et al.* [260] used a technique called Elastic Bunch Graph Matching (EBGM) where they mapped a deformable grid onto a face image by using local sets of Gabor filtered features. Numerous other methods [8, 60, 103, 143] align faces by means of approaches derived from the Active Appearance Models introduced by Cootes *et al.* [41] (see section 2.3).

Approaches belonging to the second group are less common. Moghaddam *et al.* [165], for example, used a maximum likelihood-based template matching method to eliminate translation and scale variations. In a second step, however, they detected four facial features to correct rotation as well. Jia *et al.* [117] employed a tensor-based model to super-resolve face images of low resolution, and at the same time find the best alignment by minimizing the correlation between the low-resolution image and the super-resolved one. Rowley *et al.* [207] proposed a face detection method including a Multi-Layer Perceptron (MLP) to estimate in-plane face rotation of arbitrary angle. They perform the alignment

on each candidate face location and then decide if the respective image region represents a face.

Note that some of the pose estimation methods described in the previous section also determine the in-plane rotation angle, and thus, to some extend, also these approaches can be employed for face alignment. Moreover, many facial feature detection algorithm (see section 5.2) can be used for this purpose. For example, by localizing the eyes one can correct the in-plane rotation of the face. By detecting a third feature point, *e.g.* the mouth, an affine transformation can be defined which maps these feature points onto predefined positions and thus globally aligns the face with the image borders.

4.5.3 Face Alignment with Convolutional Neural Networks

Introduction

In this section, we will propose a novel face alignment technique [82, 58] based on Convolutional Neural Networks (*c.f.* chapter 3). The method we propose is similar to the one of Rowley *et al.* [207], but it not only corrects in-plane rotation but also x/y translation and scale variations. It is further capable of treating *non-frontal* face images and employs an iterative estimation approach. After training of the CNN, it receives a mis-aligned face image and directly and simultaneously responds with the respective parameters of the transformation that the input image has undergone.

Network Architecture

The proposed neural architecture is a specific type of Neural Network consisting of seven layers, where the first layer is the input layer, the four following layers are convolutional and sub-sampling layers, and the last two layers are standard feed-forward neuron layers. The aim of the system is to learn a function that transforms an input pattern representing a mis-aligned face image into the four transformation parameters corresponding to the mis-alignment, *i.e.* x/y translation, rotation angle and scale factor. Figure 4.8 gives an overview of the architecture which is similar to LeNet-1 introduced by LeCun *et al.* [134] (*c.f.* section 3.2.2) or the Convolutional Face Finder (CFF) proposed by Garcia *et al.* [81] (*c.f.* section 4.2.3) .

The retina l_1 receives a cropped face image of 46×56 pixels, containing gray values normalized between -1 and $+1$. No further pre-processing like contrast enhancement, noise reduction or any other kind of filtering is performed.

The second layer l_2 consists of four feature maps with a receptive field size of 7×7. Each feature map performs a convolution of the input image with a different trainable kernel and adds a bias. A linear activation function is then applied to the result.

Layer l_3 sub-samples its input feature maps into maps of reduced size by locally averaging neighboring units. The size of the sub-sampling window is 2×2 and a sigmoid activation function is used here. The goal of this layer is to make the system less sensitive to small shifts, distortions and variations in scale and rotation of the input at the cost of some precision.

Layer l_4 is another convolutional layer and consists of three feature maps, each connected to two maps of the preceding layer l_3. In this layer, 5×5

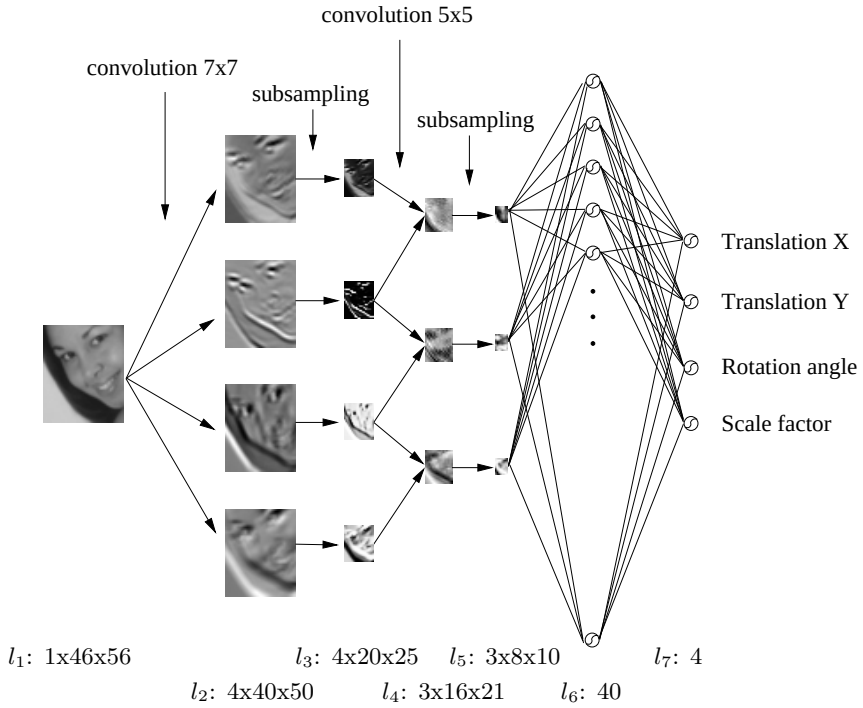


Figure 4.8: The Neural Network architecture of the proposed face alignment system

convolution kernels are used, and each feature map has two different convolution kernels, one for each input map. The results of the two convolutions as well as the bias are simply added up. The maps of layer l_4 extract higher-level features by combining the lower-level information in l_2.

Layer l_5 is again a sub-sampling layer that works the same way as l_3, *i.e.* it reduces the size of the feature maps again by a factor of two.

Whereas the previous layers act principally as feature extraction layers, layers l_6 and l_7 combine the extracted local features into a global model. They are neuron layers that are fully connected to their respective preceding layers and use a sigmoid activation function. l_7 is the output layer containing exactly four neurons, representing x and y translation, rotation angle and scale factor, normalized between -1 and $+1$ w.r.t. their minimum and maximum values $pmin_i$ and $pmax_i$, *e.g.* ± 30 degrees for the rotation angle. After activation of the network, these neurons contain the estimated normalized transformation parameters y_{7i} ($i = 1..4$) of the mis-aligned face image presented at l_1. Each final transformation parameter p_i is then calculated by linearly rescaling the corresponding y_{7i} value from [-1,+1] to the interval of the respective minimal and maximal allowed values $pmin_i$ and $pmax_i$:

$$p_i = \frac{pmax_i - pmin_i}{2}\,(y_{7i} + 1) + pmin_i \quad , \forall i = 1..4\,. \tag{4.1}$$

Figure 4.9: Examples of training images created by manually mis-aligning the face (top left: the well-aligned face image)

Training Process

We constructed a training set of about 30,000 face images extracted from several public face databases with annotated eye, nose and mouth positions. Using the annotated facial features, we are able to crop well-aligned face images where the eyes and the mouth are roughly at predefined positions while keeping a constant aspect ratio. By applying transformations on the well aligned face images, we produced a set of artificially mis-aligned face images, that we cropped from the original images and resized to the dimensions of the retina (46×56). The transformations were applied by varying translation between -6 and $+6$ pixels, rotation between -30 and $+30$ degrees and the scale factor between 0.9 and 1.1. Thus, we defined the minimal and maximal parameter values as follows:

$$\begin{aligned} pmin_1 = pmin_2 &= -6 \\ pmax_1 = pmax_2 &= +6 \\ pmin_3 &= -30 \\ pmax_3 &= +30 \\ pmin_4 &= 0.9 \\ pmax_4 &= 1.1 \end{aligned}$$

Figure 4.9 shows some training examples for one given face image. The respective transformation parameters p_i were stored for each training example and used to define the corresponding desired outputs t_i of the Neural Network by normalizing them between -1 and $+1$:

$$t_i = \frac{2 \times (p_i - pmin_i)}{pmax_i - pmin_i} - 1 \quad , \; i = 1..4 \; . \tag{4.2}$$

Training was performed using the well-known Backpropagation algorithm which has been adapted in order to account for weight sharing in the convolutional layers (l_2 and l_4) (see Alg. 8 on page 57). The objective function is simply the Mean Squared Error (MSE) between the computed outputs and the desired outputs of the four neurons in l_7 for all N training examples:

$$E = \frac{1}{4N} \sum_{k=1}^{N} \sum_{i=1}^{4} \left(y_{7i} - t_i \right)^2 \; . \tag{4.3}$$

At each iteration, a set of 1,000 face images is selected at random. Then, each face image example of this set and its known transformation parameters are

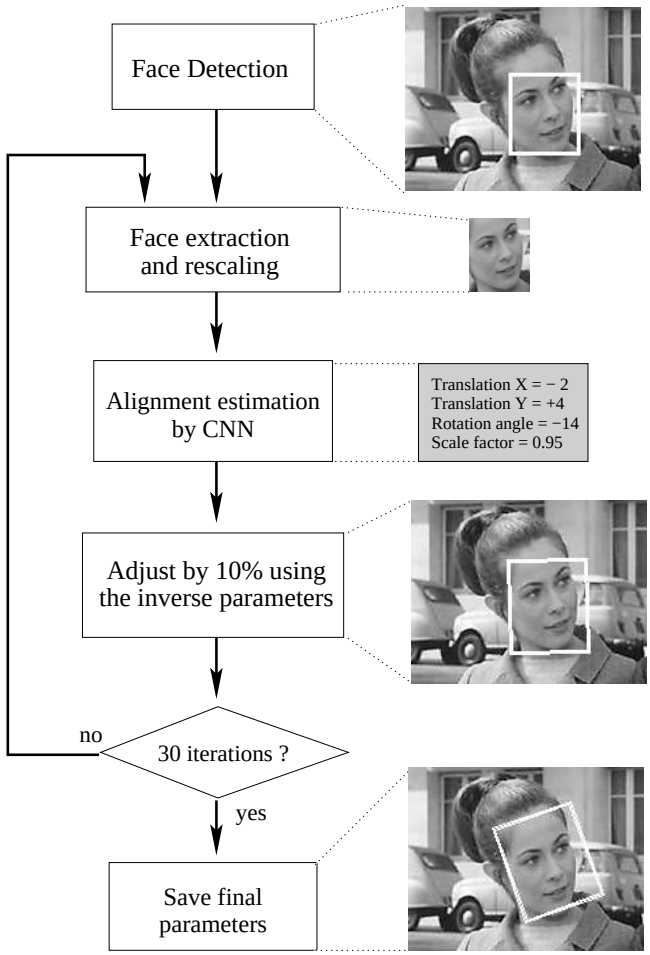


Figure 4.10: The overall face alignment procedure of the proposed system

presented, one at a time, to the Neural Network and the weights are updated accordingly (online training). Classically, in order to avoid overfitting, after each training iteration, a validation phase is performed using a separate validation set. A minimal error on the validation set is supposed to give the best generalization and the corresponding weight configuration is stored.

Alignment Process

We now explain how the Neural Network is used to align face images with bounding boxes obtained from a face detector. The overall procedure is illustrated in figure 4.10. Face detection is performed using the Convolutional Face Finder of Garcia and Delakis [81] (*c.f.* section 4.2.3) which produces upright bounding boxes. The detected faces are then extracted according to the bounding box and resized to 46×56 pixels. For each detected face, the alignment process is performed by presenting the mis-aligned cropped face image to the trained Neural Network which in turn gives an estimation of the underlying transformation.

A correction of the bounding box can then simply be achieved by applying the inverse transformation parameters ($-p_i$ for translation and rotation and $1/p_i$ for scale). However, in order to improve the correction, this step is performed several (*e.g.* 30) times in an iterative manner, where, at each iteration, only a certain proportion (*e.g.* 10%) of the correction is applied to the bounding box. At each step, the face image is re-cropped using the new bounding box and a new estimation of the parameters is calculated with this modified image. The transformation with respect to the initial bounding box is obtained by simply accumulating the respective parameters at each iteration. Using this iterative approach the system finally converges to a more precise solution than when using a full one-step correction.

The alignment precision can be further improved by successively executing the procedure described above two times with two different Neural Networks, the first one trained as presented above for coarse alignment and the second one for fine alignment. For training the fine alignment Neural Network, we built a set of face images with less variation, *i.e.* $[-2, +2]$ for x and y translations, $[-10, +10]$ for rotation and $[0.95, 1.05]$ for scale variation. The parameters $pmin_i$ and $pmax_i$ $(i = 1..4)$ change accordingly in the preceding formulas.

Experimental Results

To evaluate the proposed approach, we used two different annotated test sets: the public face database BioID [22] containing 1,520 images and a private set of about 200 images downloaded from the Internet (see Appendix A). The latter, referred to as Internet test set, contains face images of varying size, with large variations in illumination, pose, facial expressions and containing noise and partial occlusions. As described in the previous section, for each face localized by the face detector [81], we perform the alignment on the respective bounding box and calculate the precision error e which is defined as the mean of the distances between its corners a_i and the respective corners of the desired bounding box d_i normalized with respect to the width W of the desired bounding box:

$$e = \frac{1}{4\,W} \sum_{i=1}^{4} \|a_i - d_i\| \tag{4.4}$$

Figure 4.11 shows, for the two test sets, the proportion of correctly aligned faces varying the allowed error e.

For example, if we allow an error of 10% of the bounding box width, 80% and 94% of the faces of the Internet and BioID test sets respectively are well aligned. Further, for about 70% of the aligned BioID faces, e is below 5%.

We also compared our approach to a different technique that localizes the eye centers, the nose tip and the mouth center. This method is also based on CNNs and described in section 5.3. The face images with localized facial features are aligned using the same formula as used for creating the training set of the Neural Network presented in this paper. Figure 4.12 shows the respective results for the Internet test set.

To show the robustness of the proposed method, we added Gaussian noise with varying standard deviation σ to the pixels of the input images before performing face alignment. Figure 4.13 shows the error e versus σ, averaged over the whole set for both of the test sets. Note that e remains below 14% for

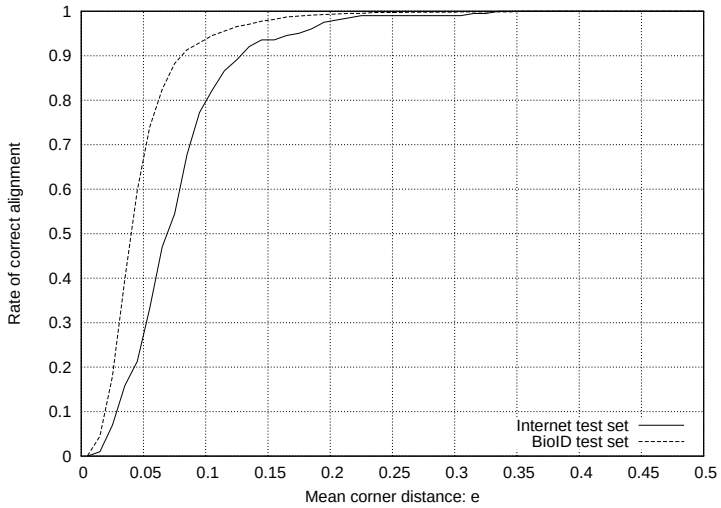


Figure 4.11: Correct alignment rate vs. allowed mean corner distance of the proposed approach

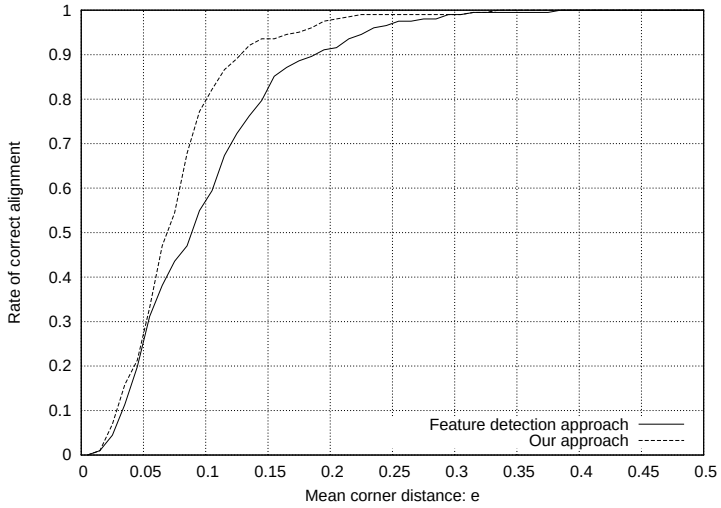


Figure 4.12: Precision of the proposed alignment approach and the approach based on facial feature detection

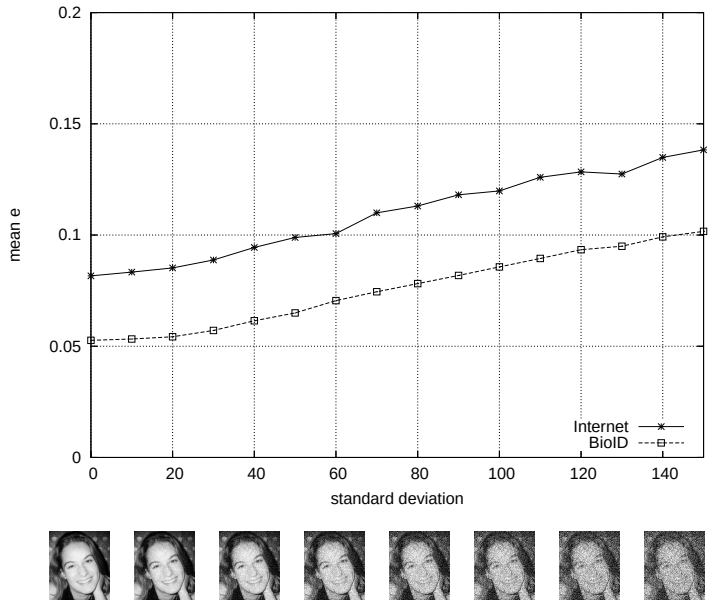


Figure 4.13: Sensitivity analysis: Gaussian noise

the Internet test set and below 10% for the BioID test set while adding a large amount of noise (*i.e.* σ up to 150 for pixel values being in $[0, 255]$).

Another experiment demonstrates the robustness of our approach against partial occlusions while adding black filled rectangles of varying area to the input images. Figure 4.14 shows, for two types of occlusions, the error e averaged over each test set with varying s, representing the occluded proportion with respect to the whole face rectangle. For a given detected face, let w be the width and h be the height of the box. The first type of occlusion is a black strip ("scarf") of width w at the bottom of the detected face box. The second type is a black box with aspect ratio w/h at a random position inside the detected box. We notice that while varying the occluded area up to 40% the alignment error does not substantially increase, especially for the scarf type. It is, however, more sensitive to random occlusions (*i.e.* the second type). Nevertheless, for $s < 30\%$ the error stays below 15% for the Internet test set and below 12% for the BioID test set.

Figure 4.15 shows some results on the Internet test set. For each example, the black box represents the desired box, while the white box on the left represents the face detector output and the one on the right the estimated aligned box.

The method is also very efficient in terms of computational time. It runs at 67 fps on a Pentium IV 3.2GHz and can easily be implemented on embedded platforms.

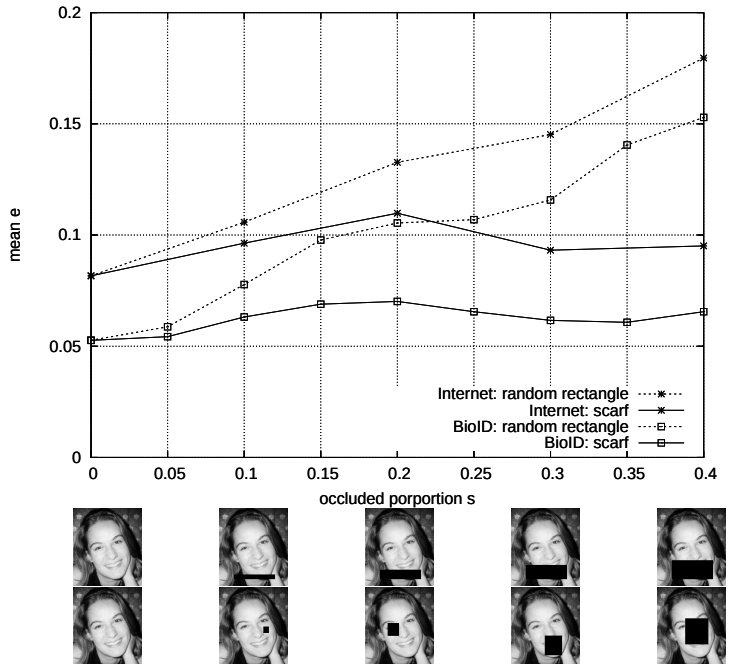


Figure 4.14: Sensitivity analysis: partial occlusion

4.6 Conclusion

In this chapter, we presented the problem of face detection in images and some of the most important approaches proposed in the literature to tackle this problem. Many face detection methods obtain relatively good results on standard public face databases, like the CMU database. However, we focused on one approach in particular, the Convolutional Face Finder (CFF) of Garcia and Delakis [81] as it is a CNN-based method and it attains a high detection rate with a low false alarm rate, which is a very desirable property in face detection. Its high performance in terms of computation time and the ability to implement it on embedded platforms are further advantages of this approach.

We then pointed out that for further face analysis tasks, notably for face recognition, a preceding *normalization* step is essential and we described possible methods employed in this context, such as illumination normalization and head pose estimation and/or correction. In this regard, we outlined some well-known approaches published in the literature.

Finally, we proposed a novel face alignment method which can be considered as a post-processing step of face detection. It is based on CNNs and trained to simultaneously estimate x/y translation as well as rotation and scale variations of the slightly mis-aligned bounding rectangles coming from a face detector. Experimental results show that this approach is precise and very robust when applied to face images under difficult illumination conditions and varying head pose. Also, a considerable robustness to pixel-noise and partial occlusions has been experimentally verified.

The combination of CFF and the face alignment system shows to perform

Figure 4.15: Some alignment results on the Internet test set. For each test image, the *black* rectangle represents the desired bounding box. The *white* rectangle on the respective *left* image shows the output of the face detector and the *white* rectangle on the *right* image shows the result of the face alignment system.

very well in real-world contexts of facial image processing applications under difficult conditions. Further, the efficient implementation, also on embedded platforms, make them a powerful technique suitable for further facial analysis such as face recognition.

Chapter 5

Facial Feature Detection

5.1 Introduction

Facial feature detection designates the problem of localizing characteristic points in face images, such as the eyes, the nose tip, the mouth, the corners of the mouth etc. Depending on the resolution of the face images being processed the localization can also imply more specific features, *e.g.* eye corners, eye brow corners, orbit points, nostrils or points marking the chin. This is sometimes also referred to as *fiducial point detection*.

Facial feature detection algorithms differ in several aspects. First of all, they can be part of a face detection method where the feature detectors scan over a larger image which may contain a face or not. Then, the presence of all or most of the facial features at a given location and a plausible constellation of them indicate the presence of a face at this particular position. Many facial feature detection algorithms, however, operate on images or image regions containing only the face which have been localized and extracted beforehand by a face detection algorithm. A further point where the algorithms differ is the type of image they are working on. This implies the image resolution as well as external conditions such as illumination, noise, background but also the range of possible head poses and facial expressions. Another important aspect is the *purpose* or the *target application* of the respective feature detection algorithm as this might impose more or less strict precision constraints as well as the type and number of features to detect. As mentioned above, they can be part of a face detection system, for example, which in turn can be part of a larger face processing system. Other possible applications are *facial expression recognition* or *face recognition* requiring a rather precise localization of specific facial features. Finally, it may be used for advanced Human-Computer Interaction (HCI) applications, *e.g.* avatar animation.

The problem of facial feature detection under unconstrained conditions is still far from being resolved, and the difficulties that remain are essentially the same as for face analysis tasks in general, notably those outlined in section 1.3, *e.g.* the sensitivity to illumination and pose variations, facial expressions and partial occlusions.

In the following, we will describe some of the most important works in this field and then present our own approach to facial feature detection using Convo-

lutional Neural Networks. Finally, we will experimentally show the performance of the proposed method in terms of detection rate and precision as well as the robustness against the types of noise mentioned above.

5.2 State-of-the-art

Numerous approaches to facial feature detection in still images have been proposed in the last decade. One could classify them into two categories: the first category applies local processing techniques to the face image, such as filters or templates, in order to obtain candidate feature positions and then selects the most likely combination (or the one with the highest score) w.r.t. a global model. Approaches belonging to the second category iteratively try to match a grid, a graph or any other form of deformable global model to a face image. The algorithms of this category are supposed to converge to a solution where the grid points correspond to the facial features to localize.

Local feature detection approaches

Vincent *et al.* [249] proposed a system detecting eye and mouth features using several trained Multi-Layer Perceptrons (MLP) for each feature to detect scanning over the input image. In a coarse-to-fine approach, they first localized the eye and mouth features and then 12 so-called "micro-features", five for each eye and two for the mouth. Finally, a maximum likelihood estimator provides the most likely combination of features.

Reinders *et al.* [194] presented a similar approach using a MLP working on gradient magnitudes and orientations in order to detect the rough eye positions and then four specialized MLPs to detect micro-features.

Yuille *et al.* [269] presented a technique based on deformable templates of the eye and the mouth. The feature localization was performed by fitting the parameterized templates to image locations where a specific energy function is minimal. The function was based on edges, peaks and valleys of the image intensity.

Other works are based on skin-color segmentation [210, 83, 229, 88], contrast enhancement and thresholding techniques [115] and/or symmetry measures [195, 111, 210] which, in practice, show to be rather sensitive to noise and illumination changes.

Leung *et al.* [141] used multi-orientation, multi-scale Gaussian derivative filters to detect feature position candidates in the input image. The best matches were retained, and for each pair of detected feature point they estimated small elliptical regions where the rest of the feature points to detect are likely to occur. In the following step, a probabilistic model based on mutual feature distances helped to identify the most likely feature positions.

Yow and Cipolla [268] also used Gaussian derivate filters to detect facial feature candidates. However, a grouping method based on belief networks was then applied to the candidate positions in order to obtain the resulting combination of facial features.

A method based on Eigen-features was presented by Moghaddam *et al.* [165]. Here, the residual error between the image patch in the search window and the reconstructed image, the so-called Distance From Feature Space (DFFS), was

calculated to obtain candidate feature positions. Finally, a maximum likelihood estimator based on Gaussian probability distributions was applied.

Extensions of this approach have been proposed by Demirel *et al.* [52] who used an additional weighting scheme based on a spatial probability distribution, and Shakunaga *et al.* [223] who applied an iterative approach using a 3D structure model.

Lin *et al.* [145] presented an approach that first segments the pixels belonging to the face by a region growing algorithm and then localizes facial features (eyes, eyebrows, nose and mouth) by a Genetic Algorithm-based search method where the fitness function uses heuristic pixel intensity measures.

A template matching method using the Discrete Cosine Transform (DCT) has been proposed by Zobel *et al.* [274]. The localization was further supported by a probabilistic framework, which they call *coupled structure*, and a Maximum A Posteriori (MAP) classifier.

Heisele *et al.* [97] proposed a template matching approach based on SVMs. In the first step, they trained one SVM on small image regions for each facial feature to detect, *i.e.* eyes, nose and mouth, and in a second step they performed a SVM classification of the geometrical configuration of the detected features.

Sirohey *et al.* [229] used empirically determined non-linear filters to detect the eye corners in skin-color segmented images. False alarms were eliminated by clustering and pairing the eye corner candidates and then applying some heuristic geometrical rules.

The approach proposed by Jesorsky *et al.* [116] consists of two steps. In order to detect a face and its rough eye positions, they first applied a Sobel filter to detect edges in the input image and then tried to fit a face edge model to it using the Hausdorff distance. In the second step, the resulting face location was refined by a smaller model that was fit to the eyes.

Garcia *et al.* [83] presented an facial feature and face detection method based on skin-color segmentation and subsequent filtering of the input image by a set of specific wavelet filters. Finally, a deformable geometric face template was applied.

Feris *et al.* [68] proposed a two-level hierarchical approach using Gabor Wavelet Networks (GWN). The first-level GWN is trained to localize the face and the rough feature positions (eye/mouth corners and nostrils). The second-level GWN then refines the localization.

Hamouz *et al.* [94] performed a detection of 10 facial features in order to localize faces. They use a bank of Gabor filters and Gaussian Mixture Models (GMM) of filter responses for the feature localization. Then, triplets of feature candidates were combined to form a set of face hypotheses. For each hypothesis, a respective normalized image patch containing the face candidate was extracted and classified as face or non-face by two Support Vector Machines (SVM) working on different image resolutions. Finally, the hypothesis, and thus the feature locations, producing the highest value of the SVM's discriminant function was selected.

The approach presented by Gourier *et al.* [88] applies a chrominance-based skin region segmentation and then detects salient facial feature points by means of first and second order Gaussian derivative filters and a k-means clustering on the filter responses. The feature positions are then obtained by a connected component analysis in the image of filter responses and some geometrical heuristics.

Iterative methods using deformable grids

Whereas the approaches of the preceding section could be denoted as *"bottom-up"*, the methods outlined in the following can be considered as *"top-down"* approaches because a global model, *i.e.* a deformable grid, is fit to the face image by some kind of iterative optimization technique operating on local appearances.

The first very well-known method of this type has been presented by Wiskott *et al.* [260] and is called Elastic Bunch Graph Matching (EBGM). As this approach is not only used for feature detection but also for face recognition it will be described in section 6.2.

More recently, Cootes *et al.* [41] introduced a technique called Active Appearance Models (AAM) (see section 2.3). By applying Principal Component Analysis (PCA) on the training set of faces with manually labeled feature positions they first constructed a statistical shape model. Then, the face images were warped to fit the mean shape and a second PCA on the so-called shape-free patches was performed in order to obtain an independent statistical texture model. To localize facial features in a new face image, the respective parameters of the shape and texture models are iteratively estimated minimizing the residual textural error between model and estimation. Note that, in order to work effectively, methods based on AAMs need a rather good initialization of feature positions.

Cristinacce and Cootes [49] proposed a method to detect 17 facial feature points. First, they applied the face detector presented by Viola and Jones [250], they resized the extracted face image to 100×100 pixels and then computed a so-called feature response image. This image contains at each location (*i.e.* pixel) the result of an Adaboost classifier combining three different feature extractors: one based on normalized eye and mouth templates, one based on orientation maps and the third using Boosted Cascade Detectors. Finally, the feature locations were determined by a non-linear optimization method, which they call Shape Optimized Search (SOS). It uses a statistical shape model, initially computed by PCA, and a fitting function taking into account both shape and the response of the feature extractors, *i.e.* the feature response image.

In another work [50], Cristinacce *et al.* proposed a feature detection technique using Boosted Cascade Detectors [250] and also a statistical model. They first localized the face and facial features with the Boosted Cascade Detectors trained on appropriate image patches. Then, they modeled the pairwise conditional distribution of feature positions using histograms, a technique which they call "Pairwise Reinforcement of Feature Responses" (PRFR). Finally, an AAM variant due to Scott *et al.* [221] was used to refine the feature positions. They showed that this method outperforms their previous work [49] as well as purely AAM-based methods [41].

Evaluation

The comparison of the performance of feature detection approaches outlined the previous sections is rather difficult as the authors used different and often private face databases to evaluate their methods. Moreover, not all methods detect the same facial features and/or the same number of features. Nevertheless, an overview of the performance of some facial feature detection methods is presented in table 5.1. The allowed error in the last column represents the error

threshold normalized w.r.t. the inter-ocular distance. A question mark signifies that the allowed error has not been reported in the paper. In this case, the detection rate does not tell very much about the precision of the respective methods.

Author	**Database**	**Facial Features**	**Detection rate**	**Allowed error**
Leung [141]	private	eyes,nose (5 pts.)	86%	?
Reinders [194]	private	right eye (4 pts.)	96%	10%
Shakunaga [223]	private	eyes,nose,mouth ears (8 pts.)	96%	?
Jesorsky [116]	BIOID [22]	eyes (2 pts.)	80%	10%
	XM2VTS [160]		92%	10%
Sirohey [229]	Aberdeen [185]	eyes (4 pts.)	90%	?
Feris [68]	Yale[16],FERET[183]	eyes (4 pts.)	95%	$\approx 5\%$
		nostrils (2 pts.)	95%	$\approx 5\%$
		mouth (2 pts.)	88%	$\approx 5\%$
Cristinacce [49]	BIOID [22]	eyes,nose,mouth (17 pts.)	85%	10%
			96%	15%
Gourier [88]	private	eyes (2 pts.)	97%	?
		mouths (1 pt.)	88.8%	?
Hamouz [94]	BIOID [22]	eyes,nose,mouth (10 pts.)	76%	5%
	XM2VTS [160]		88%	5%
	BANCA [7]		81.4%	5%

Table 5.1: Overview of detection rates of some published facial feature detection methods

For an allowed error of about 5%, the approach proposed by Feris *et al.* [68] shows good results on set of images from the Yale [16] and FERET [183] databases. And for an allowed error of 10%, Jesorsky *et al.* [116] obtain acceptable results on the BioID [22] and the XM2VTS [160] databases. Reinders *et al.* [194] also report a good detection rate. However, their method only detects eye features, and unfortunately they used a private database. Finally, very good results have been obtained by the approach proposed by Cristinacce and Cootes [49], *i.e.* 96% with 15% allowed error and 85% with 10% allowed error, considering the rather difficult database BioID they used. Refer to Appendix A for excerpts from some of these databases.

Up to the present, most of the published approaches have been evaluated on face databases containing images taken under rather constrained conditions. In contrast to these methods, we will propose a novel system based on Convolutional Neural Networks that is able to precisely and robustly detect facial features under less constrained conditions.

5.3 Facial Feature Detection with Convolutional Neural Networks

5.3.1 Introduction

We propose a hierarchical neural-based facial feature detection scheme which is designed to precisely locate fine facial features in faces of variable size and appearance, rotated up to ±30 degrees in the image plane and turned up to ±60 degrees, in complex still gray-scale images [53, 54, 82, 59]. Using still gray-scale images makes the approach more generic but also more challenging as skin color or motion clue cannot be used to aid detection. The system we present is able to detect facial features in very complex images taken under a wide variety of poses, in low resolution images, with partial occlusions, noise and extreme lighting variations. The proposed approach is divided into three stages; in the first stage, faces contained in the image are automatically located. The second stage consists in finding the approximate positions of local facial features within the detected face regions. In a final stage, these feature positions and their surrounding image regions are used to locate finer facial features.

The face detector as well as the coarse and fine feature detectors are based on a specific architecture of convolutional and hetero-associative neural layers. They consist of several different neural network components forming a pipeline of specialized image transformations, learned automatically from a training set of faces with annotated facial features. All components of a feature detector are sequentially connected and can thus be trained by simply presenting the input image and desired output, *i.e.* feature positions. Global constraints or models are automatically learned and used implicitly in the detection process. After training, the facial feature detection system acts like a pipeline of simple convolution and sub-sampling modules that treat the raw input face image as a whole and build facial feature maps where facial feature positions are easily retrieved.

5.3.2 Architecture of the Facial Feature Detection System

The proposed system consists of a number of independent detection components operating in a hierarchical way and on different parts of the image (see Fig. 5.1). The first component performs automatic face detection using the "Convolutional Face Finder" (CFF) proposed by Garcia and Delakis [81] (see section 4.2.3) which detects faces and finds the respective detected face bounding boxes in the input images.

The proposed *Facial Feature Detector* (FFD) processes the previously extracted face regions and locates four facial features: the eye centers, the tip of the nose and the center of the mouth. Then, the eye and mouth regions are processed by a specialized *Eye Feature Detector* (EFD) and a *Mouth Feature Detector* (MFD) which locate finer facial features; the EFD is designed to detect the eye corners and the pupil, and the MFD is designed to detect the left and right corner of the mouth and the center points on the outer edges of the upper and lower lip.

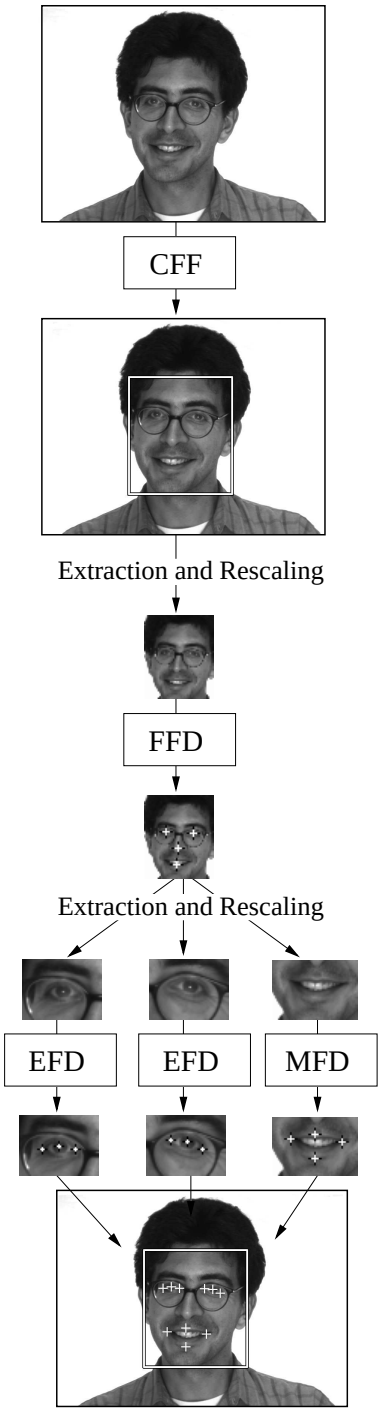


Figure 5.1: Principal stages of the feature detection process of the proposed approach

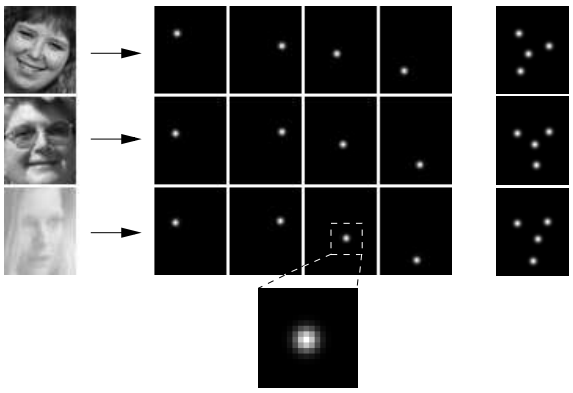


Figure 5.2: Some input images and corresponding desired output feature maps (right eye, left eye, nose tip and mouth center). For illustration purposes, the last column shows the superposition of the desired outputs and the image at the bottom shows an enlarged area of one of the desired feature maps

The Initial Facial Feature Detector

The architecture of the initial Facial Feature Detector (FFD) is a specific type of Convolutional Neural Network (CNN) (*c.f.* chapter 3) consisting of six layers, where the first layer is the input layer, the three following layers are convolutional and sub-sampling layers, and the last two layers are standard feed-forward neuron layers. The aim of the system is to learn a non-linear function that transforms a raw input face image into desired output feature maps where facial features are highlighted (see Fig. 5.2).

Figure 5.3 gives an overview of the FFD architecture. The retina l_1 receives a cropped face image of 46×56 pixels, containing gray values normalized between -1 and $+1$. No further pre-processing such as contrast enhancement, noise reduction or any other kind of filtering is performed by the retina. The second layer l_2 consists of four feature maps each having a 7×7 trainable kernel and a linear activation function. Layer l_3 is a sub-sampling layer with window size 2×2 and sigmoid activation function. Layer l_4 is another convolutional layer and consists of only one feature map with a 5×5 kernel and sigmoid activation function.

While the previous layers act principally as feature extraction layers, layers l_5 and l_6 use the local information to form a global model. Layer l_5 is composed of sigmoidal neurons fully connected to layer l_4 and is dedicated to learning models (or constellations) of features and to activate the targeted positions in the output feature maps. This part of the network was inspired by *Auto-Associative* Neural Networks (AANN) (*c.f.* section 2.8.4) which are trained to reproduce an input (pattern) by means of a hidden layer containing much less neurons than the input dimension. It has been shown that AANNs effectively perform a dimensionality reduction similar to the one produced by a Principal Component Analysis (PCA). In our case, we do not want to *reproduce* the input but rather to *associate* the output of the feature map in l_4 with the desired output in l_6. In that way, we only allow certain constellations of features in layer l_6 to be activated. This global processing step makes the system less

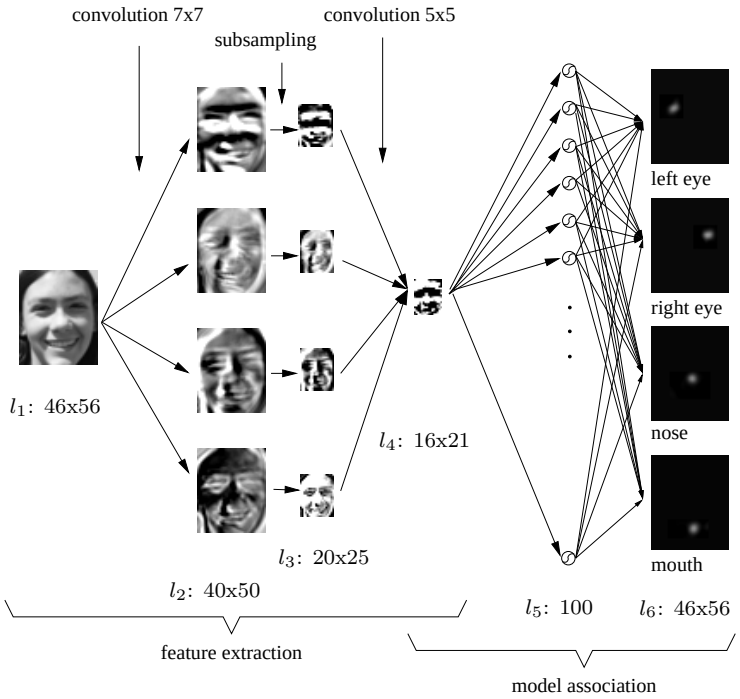


Figure 5.3: Architecture of the facial feature detector FFD

sensitive to partial occlusions and noise, *e.g.* if one eye is not visible, its position is inferred by the positions of other visible local features by activating the most likely constellation. Indeed, each of the four feature maps is activated via a set of learned weights which receive a global description of the face produced in layer l_4. Layer l_5 contains 100 neurons, and the output layer l_6 is composed of four feature maps, one for each feature that is to be detected. These maps have the same dimensions as the image at the input layer, *i.e.* 46×56, and are fully connected to the preceding neurons. Sigmoid activation functions are used for both layers. In sections 5.3.3 and 5.3.4, we will describe the training methodology and the way in which this neural architecture can be efficiently applied for automatic facial feature detection.

Finer Facial Feature Detectors

As previously mentioned, we designed two types of specialized feature detectors: the Eye Feature Detector (EFD) and the Mouth Feature Detector (MFD). The EFD is trained to detect the eye corners and pupils in eye images and the MFD to detect mouth corners and upper/lower lip boundaries in mouth images. Figures 5.4 and 5.5 show examples of images with desired facial feature points, the desired output feature maps and the superposed desired output maps.

The architecture of the previously described FFD and the EFD/MFD architectures differ only in the number of feature maps and neurons in the respective layers. The number of layers, the type of neurons in the respective layers and the connection scheme are the same for all feature detectors. The components of the EFD and the MFD are as follows:

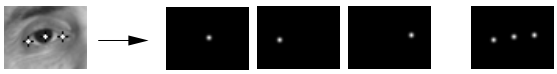


Figure 5.4: Example of an EFD input image with desired facial feature points, desired output maps and superposed desired output maps.

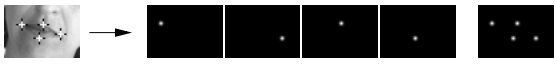


Figure 5.5: Example of a MFD input image with desired facial feature points, desired output maps and superposed desired output maps.

- EFD: l_1:53x40, l_2:4x47x34, l_3:4x23x17, l_4:19x13, l_5:50, l_6:3x53x40
- MFD: l_1:57x39, l_2:5x51x33, l_3:5x25x16, l_4:21x12, l_5:120, l_6:4x57x39

The different parameters governing the proposed architectures, *i.e.* the number of layers, the number of planes and their connectivity, as well as the size of the receptive fields, have been chosen experimentally. Practically, different architectures have been iteratively built, trained, and tested over large training sets. We retained the architecture that performed efficiently in terms of good detection rates while still containing an acceptable number of free parameters.

5.3.3 Training the Facial Feature Detectors

Each of the different parts of the system, *i.e.* CFF, FFD, EFD and MFD, is trained separately with a specific training set. CFF training is described in section 4.5.3. For eye feature detection, a single EFD is trained to treat left eye images; right eye features can then be detected by simply mirroring the input eye image horizontally. FFD, EFD and MFD are all trained using the same procedure.

The FFD training data set we used consists of face images extracted from the following databases: FERET [183] (744 images), PIE [227] (1,216 images), the Yale face database [16] (165 images), the Stirling face database [185] (185 images) as well as some face images downloaded from the Internet (167 images) (see Appendix A). In total, it comprises about 2,000 face images used for training and 500 face images used for validation, each of them centered and normalized in scale. The EFD and MFD training data consists of eye and mouth images extracted from Yale (EFD: 290 images; MFD: 290 images), AR [154] (EFD: 670 images; MFD: 375 images) and also images downloaded from the Internet (EFD: 172 images; MFD: 88 images), which results in a total number of 1,132 eye images and 753 mouth images. In order to make the system more robust to translation, rotation and scale, we created virtual samples of the extracted images by applying small translations (-2 and $+2$ pixels), rotation (from -20 to $+20$ degrees with steps of 5 degrees) and scaling (by a factor of 0.9 and 1.1). Figure 5.6 shows one of the training images and the respective transformed images. At the start of the training phase, the respective desired output maps are built to contain the value $+1$ at the feature positions and -1 everywhere else. However, in order to improve convergence and robustness, we want these output values to decrease smoothly in the neighborhood of each

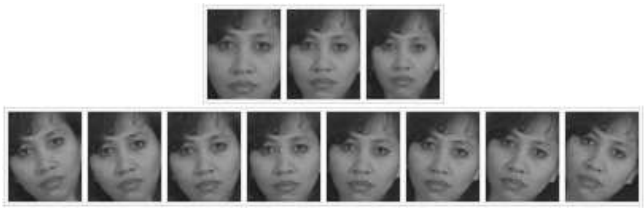


Figure 5.6: Virtual face images created by applying various geometric transformations

feature position. Therefore, the desired output maps are created using two-dimensional Gaussian functions centered at the feature position and normalized between -1 and $+1$. Thus, for a particular desired feature map o having its desired feature at position (μ_x, μ_y), the function is as follows:

$$o(x,y) = 2e^{-\frac{1}{2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}\right)} - 1$$

In our experiments, we set the variances σ_x^2 and σ_y^2 to 2. For illustration purposes, Fig. 5.2 shows an enlarged part of a desired feature map around the respective feature position.

The training phase was performed using the standard Backpropagation algorithm with momentum which has been adapted in order to account for weight sharing in the convolution layers l_2 and l_4 (see Alg. 8 on page 57). At each iteration, every image of the training set is presented to the system and the weights are updated accordingly (online training). Classically, in order to avoid over-fitting, after each training iteration, a validation phase is performed using the validation set. In fact, a minimal error on the validation set is supposed to provide best generalization and the corresponding weight configuration is stored. We considered two alternative error criteria to minimize:

- the mean-squared error (MSE) between the values of the output maps $y_k(i,j)$ and the respective values of the desired output maps $o_k(i,j)$, *i.e.* the error is calculated neuron by neuron:

$$E_{MSE} = \frac{1}{FNM}\sqrt{\sum_{k=1}^{F}\sum_{i=1}^{N}\sum_{j=1}^{M}(y_k(i,j)-o_k(i,j))^2}\,, \tag{5.1}$$

  where F is the number of output feature maps and $N \times M$ their dimension.

- a kind of distance between the detected output feature positions (i_{yk}, j_{yk}) and the respective desired output feature positions (i_{ok}, j_{ok}):

$$E_{dist} = \frac{1}{F}\sqrt{\sum_{k=1}^{F}\left((i_{yk}-i_{ok})^2+(j_{yk}-j_{ok})^2\right)}\,, \tag{5.2}$$

  where

$$(i_{yk}, j_{yk}) = \underset{i,j}{\operatorname{argmax}}\, y_k(i,j) \quad \text{and} \tag{5.3}$$

$$(i_{ok}, j_{ok}) = \underset{i,j}{\operatorname{argmax}}\, o_k(i,j) \tag{5.4}$$

In our experiments, using E_{dist} provided the best results in terms of precision on the test sets.

5.3.4 Facial Feature Detection Procedure

Figure 5.1 illustrates the principal stages of the proposed feature detection system. In the first step, the CFF detects faces in the input image and outputs the respective face bounding boxes. The extracted faces are resized to the retina size of the FFD and processed by the trained FFD. The eye positions, the nose tip and the mouth center in the resized face image can be inferred directly by simply searching for the global maximum in each of the four output maps. As the face bounding boxes may be imprecise, the last steps are repeated for slightly translated and scaled face image regions. In our experiments, we achieved good results with translations by $-4, -2, 0, +2, +4$ pixels and scale factors of $0.9, 1.0$ and 1.1. Then, for each face image region, the sum of the maxima of the output maps is taken as a confidence measure, and the solution having the maximal sum is selected. Having located eye and mouth positions, the eye and mouth regions of the original image are extracted, resized and passed on to the trained EFD and MFD. Eye and mouth features are then detected by EFD and MFD in the same way as FFD detects the four initial facial features. Finally, the detected feature positions are mapped onto the original image.

5.3.5 Experimental Results

In order to measure the performance of the proposed facial feature detector system, we first conducted several tests on FFD separately using pre-cropped face images, and later on for the whole feature detection system, *i.e.* CFF, FFD, EFD, MFD. To this end, we created several test sets with annotated images that are neither contained in the training nor in the validation set. The test face images for FFD were extracted from PIE (1,226 images), FERET (1,058 images) and from images from the Internet (384 images) (see Appendix A). As for the training and validation sets, the test sets were augmented with small transformations of the original images, *i.e.* translation, rotation and scaling.

The test images were presented to the feature detectors, and for each image the mean Euclidian distance m_e between the n detected feature positions and the true feature positions, normalized with respect to the inter-ocular distance d_{eyes}, is computed:

$$m_e = \frac{1}{n\, d_{eyes}} \sum_{k=1}^{n} \sqrt{(i_{yk} - i_{ok})^2 + (j_{yk} - j_{ok})^2}\ , \tag{5.5}$$

where (i_{yk}, j_{yk}) is the position output by the system, and (i_{ok}, j_{ok}) is the true position of feature i. Over all processed images of size 46×56 pixels, d_{eyes} varies between 11.0 and 23.7 pixels. Thus, for example an error of 5% of the inter-ocular distance represents about 0.5 to 1.2 pixels; an error of 10% represents 1.1 to 2.37 pixels. Note also that the manual feature annotation of the training data can entail small errors which we neglect in these estimations.

The analysis of the performance of the FFD is presented in Figs. 5.7-5.16. Figure 5.7 shows the proportion of faces with successfully detected features varying a maximum m_e allowed. The FERET test set clearly gave the best

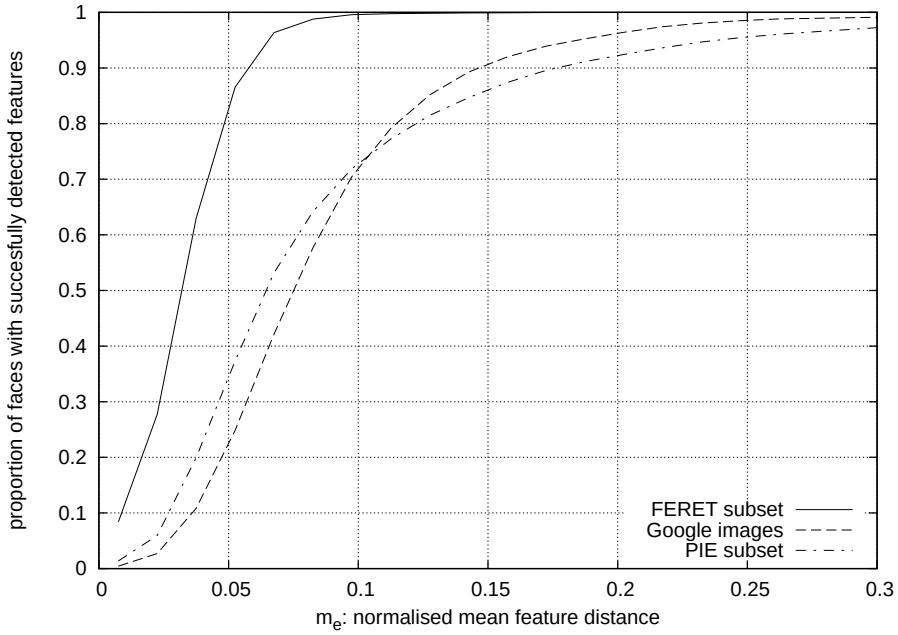


Figure 5.7: Detection rate versus m_e of the four features

detection results because there are practically no pose and lighting variations as opposed to the PIE test set. The detection rate is 99.6% if an error m_e of 10% of the inter-ocular distance (around 1.7 pixel) is tolerated for the FERET test set. The Google test set additionally contains images of low quality, with noise, extreme lighting and pose variations and partial occlusions. For the PIE and Google test sets the good detection rate is around 72% if an error m_e of 10% of the inter-ocular distance is tolerated and over 92% with m_e=20% (mean error of around 3.4 pixels).

Figures 5.8-5.10 show, for FERET, the Google images and the PIE subset respectively, the proportion of successfully detected features for each of the four features separately while varying m_{ei}. Note that the jagged appearance of the curves is due to the relatively low resolution of the images, *i.e.* distance measures are discretized very roughly. Obviously, the detection of the eyes is more precise than the detection of the tip of the nose and the mouth. Clearly, this is due to the fact that the local appearance of the eyes varies less under different poses and lighting conditions. The detection results of the tip of the nose are the least reliable. This seems plausible because the PIE test set, for example, shows considerable variations in pose and lighting and thus a large variation in the appearance of the nose. The mouth is also subject to strong variations due to facial expressions (*e.g.* smile, open/closed mouth).

The sensitivity of the proposed algorithm regarding extreme lighting and pose variations may be reduced by preprocessing the images before feature detection, using explicit models of face illumination and pose (see sections 4.3 and 4.4). In the proposed approach, however, we considered learning the possible illumination and pose variations without explicitely modeling them. Finding a way to couple the Neural Network with some a-priori physics-based illumination

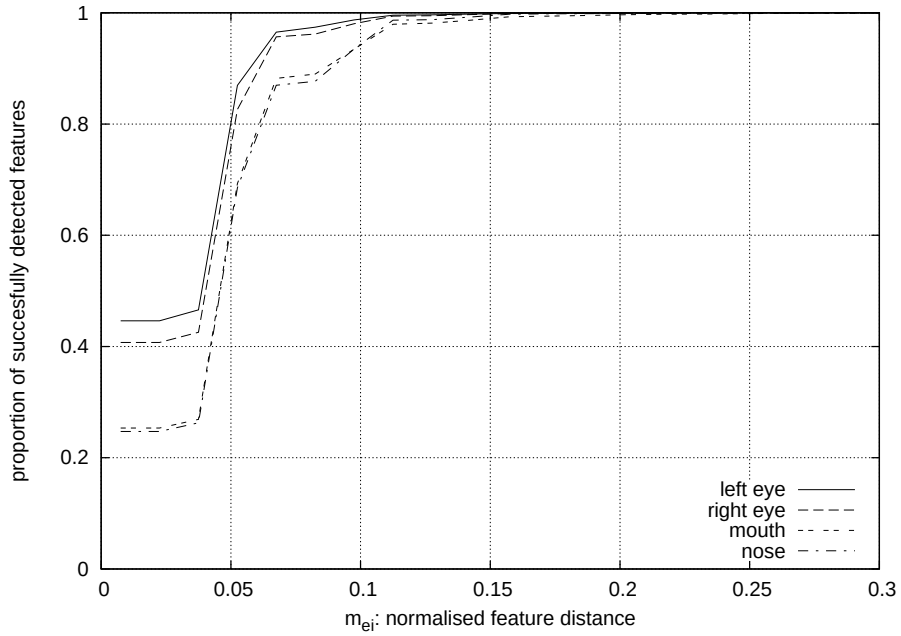


Figure 5.8: Detection rate versus m_{ei} of each facial feature (FERET)

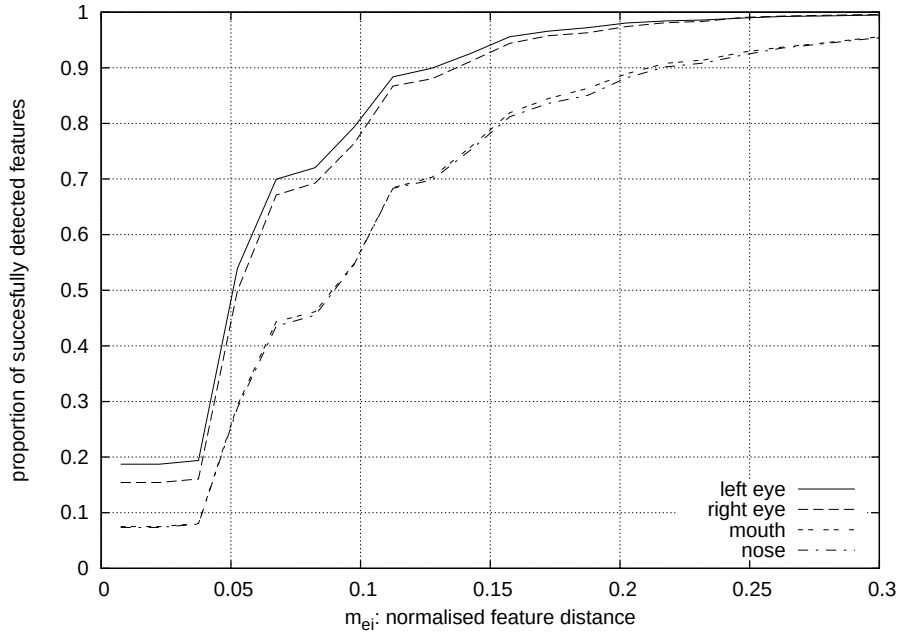


Figure 5.9: Detection rate versus m_{ei} of each facial feature (Google images)

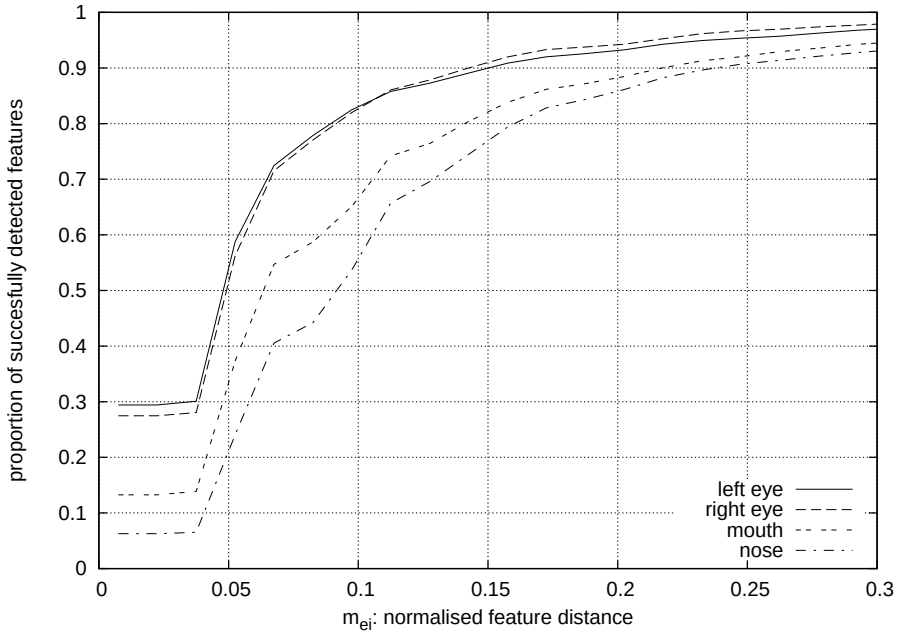


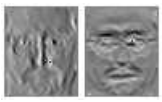
Figure 5.10: Detection rate versus m_{ei} of each facial feature (PIE subset)

models may reduce the degrees of freedom of the model and help classification. This is an issue which is not treated in this work but might be an interesting idea to explore in the future.

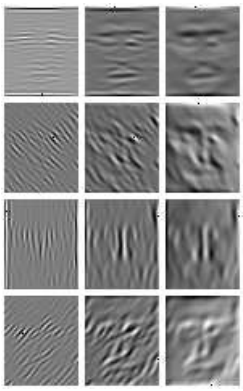
So far, we simply used the normalized *pixel intensities*, *i.e.* the raw images, as the CNN input. Figure 5.11(a) shows one example image. It is however not obvious that this is the optimal choice of input features. For that reason, we also trained two other CNNs with the same training data but receiving pre-processed face images at their input layer. The first alternative is to use *gradient images*. In this case, the input images are processed with a horizontal and a vertical Sobel filter resulting in two gradient images. Thus, the input layer now contains two input maps fully connected to the first convolution layer. Figure 5.11(b) illustrates the respective filtered results of the image shown in 5.11(a). The second alternative is to process the input face image with *Gabor wavelet filters* of selected scales and orientations. Gabor wavelets are hierarchically arranged, Gaussian modulated sinusoids. There is evidence that this is the way images are processed in the primary visual cortex (V1) of the human brain [150]. We chose three different scales and four orientations resulting in 12 Gabor wavelets. Figure 5.11(c) shows the resulting outputs of the respective wavelet filters for the image in Fig. 5.11(a). Note that in any case the input images are normalized to be in the range $[-1, +1]$ before presenting them to the CNN.

We trained two distinct CNNs with the same training data as the previously described FFD system (see section 5.3.3), *i.e.* for the task of detecting four feature points. The architecture of the two CNNs was almost the same as the one depicted in Fig. 5.3. The only difference was the number of inputs depending on the type of features used, *i.e.* two input maps for gradient images and 12 maps for the Gabor wavelet filter responses.

(a) Raw input image



(b) Gradient images



(c) Gabor wavelet filter responses

Figure 5.11: The different types of input features that have been tested

Figures 5.12-5.14 compare the precision of the three trained CNNs by means of the ROC curves on the FERET database, the Google images and the PIE subset respectively. The ROC curves clearly show that on all tested databases the CNN trained with the raw input images, *i.e.* only image intensities, outperforms those trained with gradient images and Gabor wavelet filter responses. These results suggest that CNNs trained for this type of face analysis task somehow make use of the information contained in the absolute image intensities which is largely or completely suppressed by using gradients or Gabor wavelet filters.

Another negative side-effect of using gradient images and especially Gabor wavelet filtered input images is the higher number of connections and thus trainable parameters of the CNN. This increase in complexity not only entails a higher risk of overfitting the data but also a much slower learning and processing time of the CNN in addition to the time spent for filtering the input images. Finally, when using Gabor wavelets one has to "manually" choose appropriate scales and orientations of the respective filters for a given problem, which is not a trivial task. Hence, for the rest of this work, we only use the raw images as the input for the CNNs.

Let us now analyze the sensitivity of the proposed feature detection system with respect to pixel noise and partial occlusions. To this end, we conducted two experiments on the FERET test set, the PIE subset and the Google images. In the first experiment, we added Gaussian noise with varying standard deviation σ to each pixel of the normalized face images (being in the range [0..255]). Figure 5.15 shows the mean feature error m_e with σ varying from 0 to 70. Note that the proposed feature detector is very robust to noise as the error m_e remains rather low while adding a considerable amount of noise. For the worst of the three test sets (PIE subset) m_e stays below 20% for $\sigma = 50$. Example images with corresponding amounts of added noises are shown at the top of Fig. 5.15.

The second experiment consists in occluding a certain percentage of the face images by a black zone in the lower part. Figure 5.16 shows the mean feature error m_e with an occlusion from 0 to 50%. For occlusions smaller than 40% the only invisible feature, in most of the cases, is the mouth, and the error m_e remains almost constant. Larger occlusions cover both mouth and nose, which explains the abrupt increase of m_e in all of the three test sets.

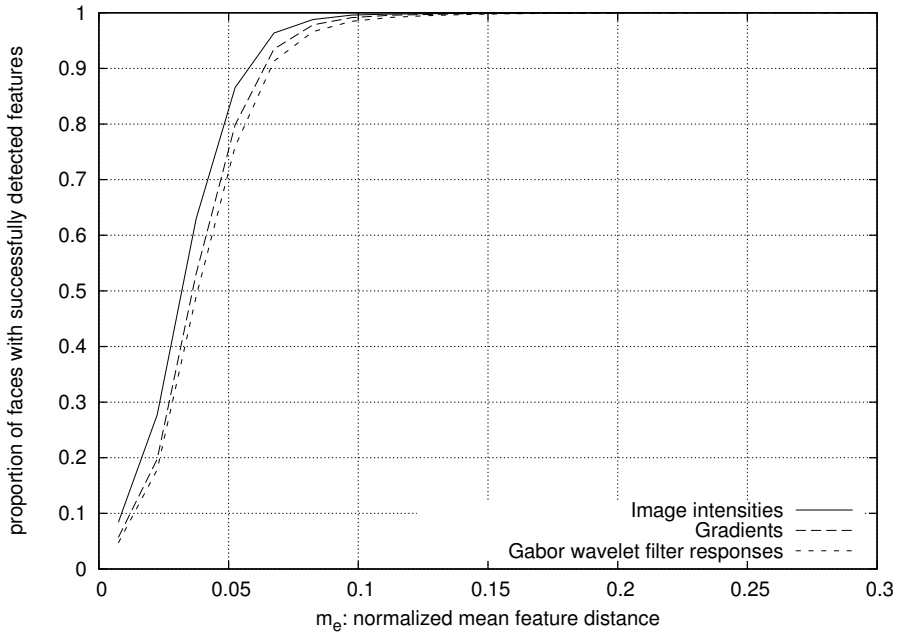


Figure 5.12: ROC curves comparing the CNNs trained with different input features (FERET database)

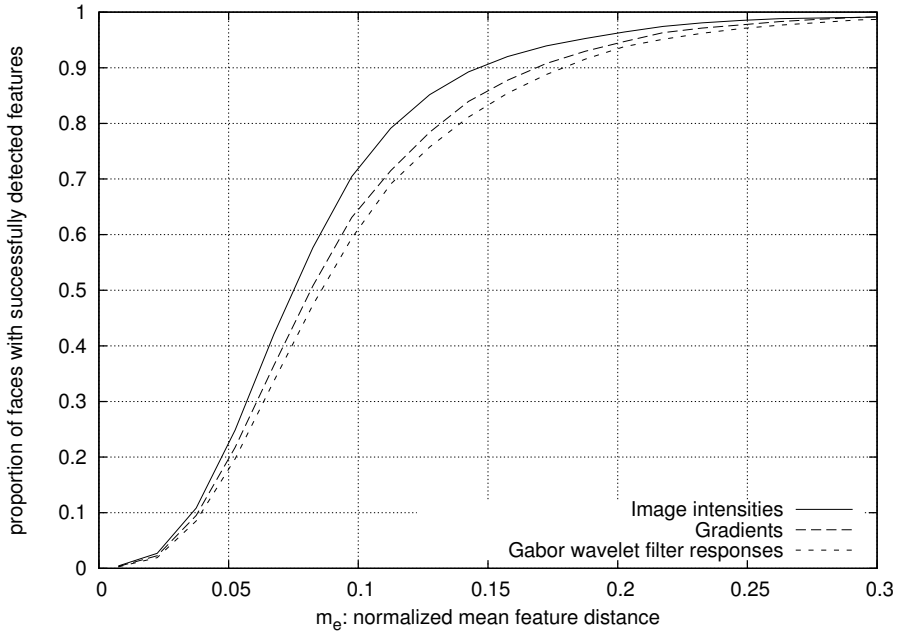


Figure 5.13: ROC curves comparing the CNNs trained with different input features (Google images)

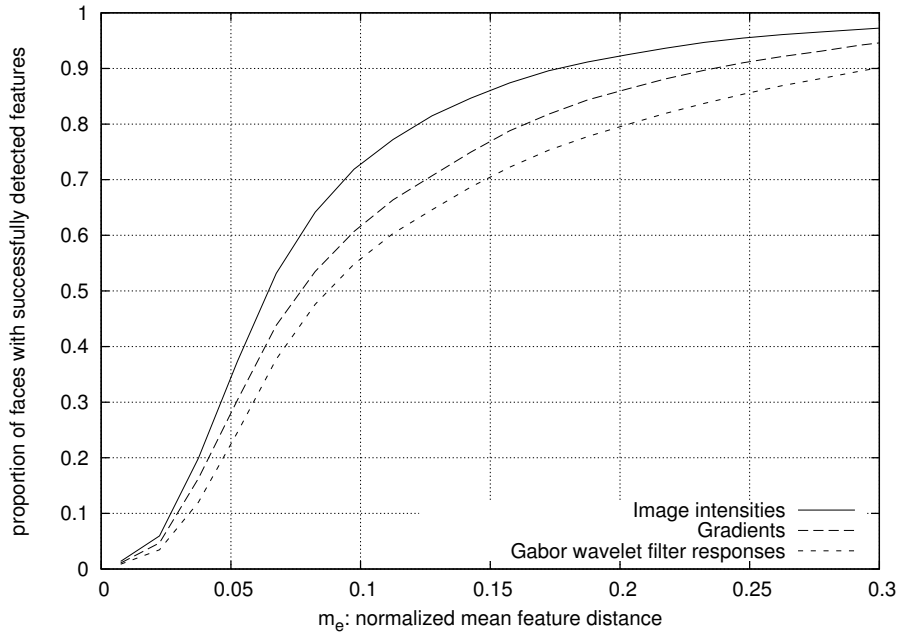


Figure 5.14: ROC curves comparing the CNNs trained with different input features (PIE subset)

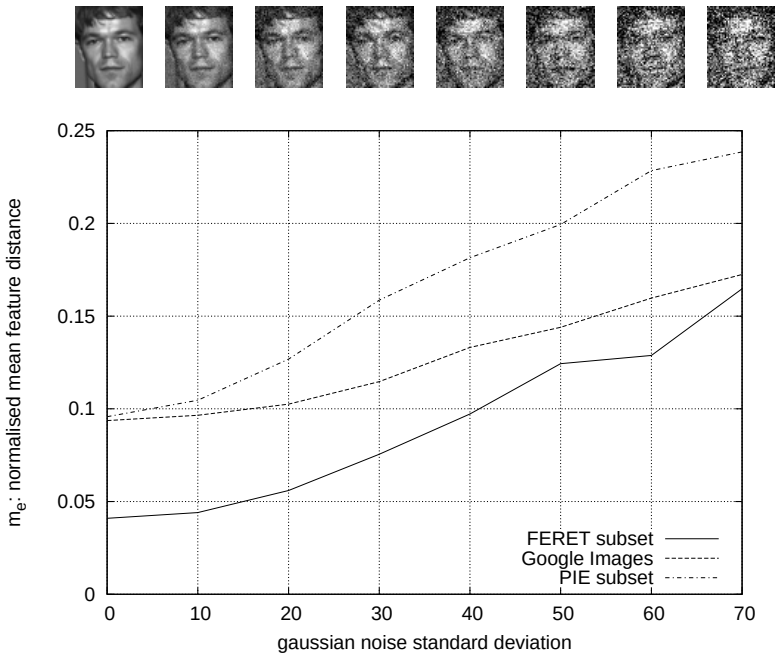


Figure 5.15: Sensitivity analysis: Gaussian noise

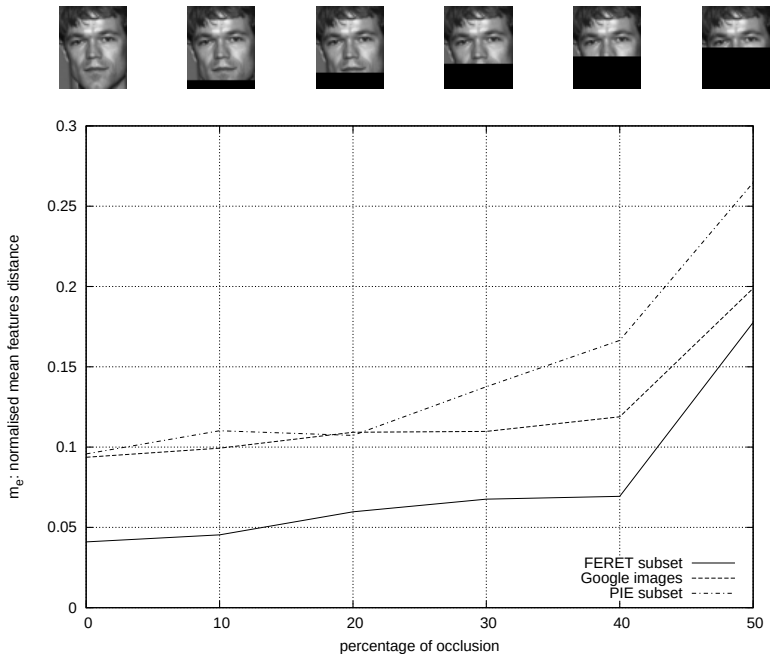


Figure 5.16: Sensitivity analysis: partial occlusion

Finally, we tested the performance of the whole feature detection system as described in section 5.3.4. Figure 5.17 shows some example images of the FERET, BioID and the AR face database with the 10 detected feature points. The databases contain frontal-view face images with different illumination and facial expressions.

Precision results for the AR and BioID databases are illustrated in Fig. 5.18 and Fig. 5.19 respectively. Note that none of the test images has been used for training the system. The figures show the detection rates for eye features (6 points) and mouth features (4 points) separately as well as the overall detection rate, *i.e.* for all 10 points. Note that the overall precision of eye and mouth feature positions is higher than the initial feature positions detected by the FFD. The detection rate of the complete system on the AR face database is 96% and 87% on the BioID database if an error m_e of 10% of the inter-ocular distance is tolerated.

If we only consider the precision of the eye pupils, we can compare our results to some other published methods. Table 5.2 summarizes the detection rates on the BioID database. The proposed approach clearly outperforms the methods proposed by Jesorsky *et al.* [116] and Hamouz *et al.* [93]. The approach proposed by Cristinacce *et al.* [50] performs slightly better for an allowed error of 10%, but its detection rate is clearly inferior for a smaller allowed error (5%) and also for an allowed error of 15%.

Figure 5.20 shows some results obtained on various images of the Google test set. The images are of rather low quality or contain faces in varying poses and under difficult lighting conditions. Notice that the system is able to cope with varying head pose, partial face occlusions (black glasses, occluding objects), extreme illumination and very poor resolution images. The last line in

Figure 5.17: Facial feature detection results on different face databases: *top:* FERET, *middle:* BioID, *bottom:* AR face database

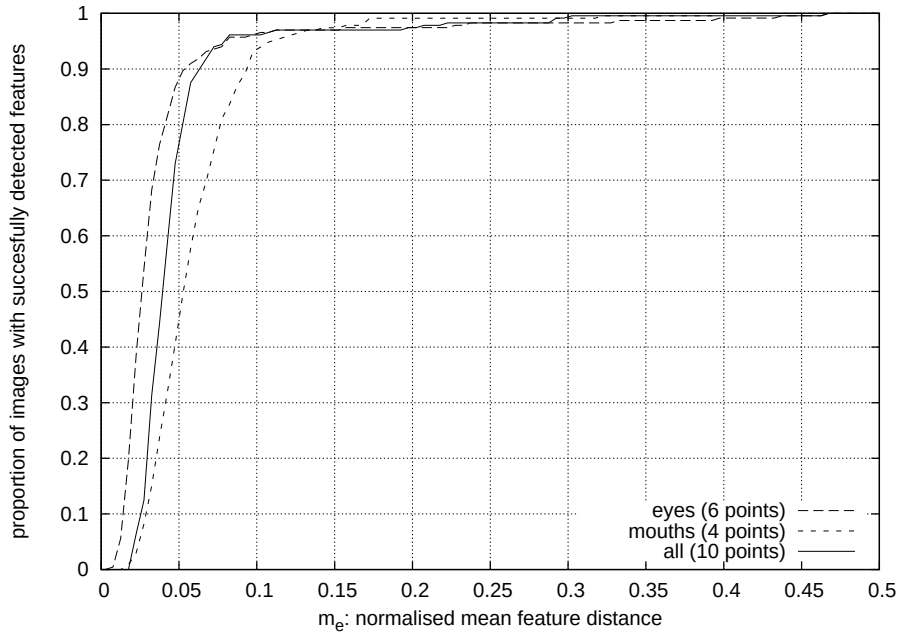


Figure 5.18: Overall detection rate for AR

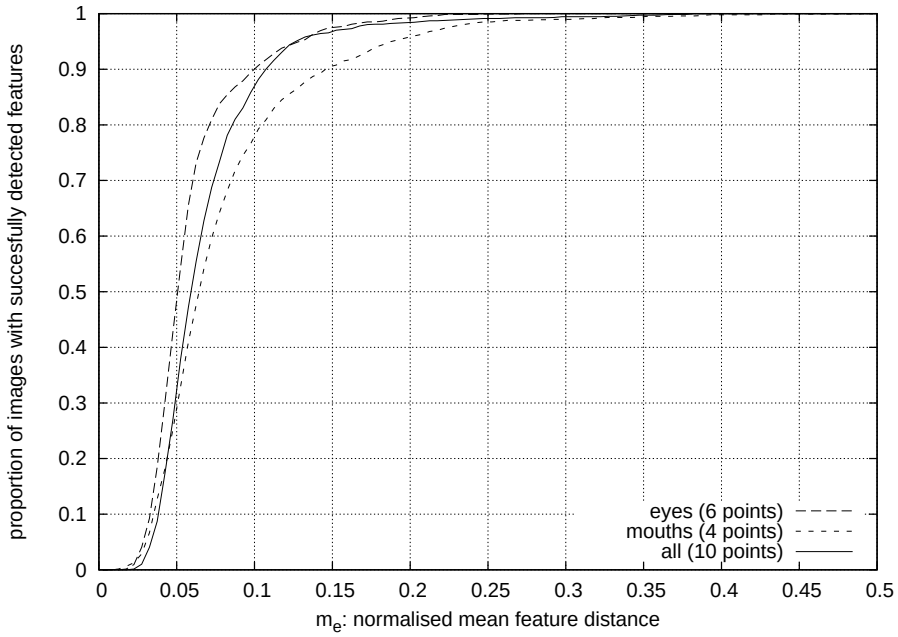


Figure 5.19: Overall detection rate for BioID

	Allowed error w.r.t. d_{eyes}		
Method	5%	10%	15%
Jesorsky *et al.* [116]	40%	79%	93%
Cristinacce *et al.* [50]	60%	96%	97%
Hamouz *et al.* [93]	50%	66%	70%
proposed approach	79%	92%	98%

Table 5.2: Comparison of eye pupil detection rates of some published methods on the BioID database

Figure 5.20: Some results of combined face and facial feature detection

Fig. 5.20 contains some images with imprecisely detected features. Naturally, the precision drops with this test set but we still obtained a detection rate of 76% with an allowed error of 10% and 92% with an allowed error of 15% of the inter-ocular distance.

The FFD detecting four facial feature points has also been implemented on embedded platforms by Roux *et al.* [204]. A considerable gain in speed of the algorithm has been obtained by the authors by reducing the memory usage and conducting further algorithmic optimizations. The final version processed 68.7 faces/s on a Pentium IV at 3.2 GHz and 12.8 faces/s on a Xscale PXA27x at 624 MHz. For an application with integrated face and facial feature detector on a Xscale PXA27x, they further report an execution time of 7 frames/s for QCIF images (176×144 pixels) containing one face per image.

5.4 Conclusion

In this chapter, we presented the problem of facial feature detection in images, which is considered a key issue in many facial analysis applications. We first outlined the state-of-the-art in this field and then proposed a novel approach to facial feature detection using Convolutional Neural Networks (CNN). The system is composed of several CNNs having a specific architecture and operating on different levels in a hierarchical manner. The first level CNN which we called FFD detects four facial features: the eyes, the nose tip and the mouth center and then extracts the regions around the eyes and the mouth and passes them on to specialized feature detectors, the so-called Eye Feature Detector (EFD) and the Mouth Feature Detector (MFD), which in turn detect 10 more specific facial features such as the eye or mouth corners.

Each of the single feature detectors is trained with a set of face, eye or mouth images annotated with the respective feature positions.

Our experimental results show that using the image intensities as input of the CNN yields to the best results compared to gradient images and Gabor wavelet filter responses. We then presented the detection rates on different public and private face databases and experimentally show that the proposed approach is precise and very robust to pixel noise and partial occlusions. Further, the system shows to be insensitive to extreme lighting conditions and varying pose. Comparing the precision results of eye pupil detection on the BioID database, the proposed approach outperforms other state-of-the-art systems.

Chapter 6

Face and Gender Recognition

6.1 Introduction

Automatic face recognition is a central topic in face analysis-related research, and it has gained a lot of importance during the last decades due to the vast number of possible applications (see section 1.2). Let us recall that one can differentiate two types of applications: *open-world* and *closed-world*. A *closed-world* face recognition system only copes with a limited number of *known* persons whereas an *open-world* application also implements the notion of an *unknown* person, corresponding to all persons that have not been explicitely learned by the system. Moreover, as already mentioned in section 1.2, face recognition can be employed for *authentication* or *verification* purposes. That means, given an input face image, a face *authentication* system is supposed to reply with the identity of the respective person, whereas a face *verification* system additionally receives a claimed identity and decides if this identity corresponds to the face or not. In any case, both of these applications are classification problems, and, from a technical point of view, they differ only marginally.

In this chapter, we will briefly describe some of the most important approaches to automatic face recognition. We will then propose a novel face recognition method based on image reconstruction using Convolutional Neural Networks. Further, we present experimental results showing the effectiveness of this approach and its robustness to noise and partial occlusions.

Finally, we will turn to the problem of *gender* recognition which is similar to face recognition in that it is also a classification problem. However, in this case, there are only two classes: male and female. We propose a CNN-based approach with a topology similar to the Convolutional Face Finder (CFF) described in section 4.2.3. We then experimentally show that, in terms of classification rate, this classifier achieves comparable results to previously published gender recognition methods based on Adaboost and SVMs.

6.2 State-of-the-art in Face Recognition

There are numerous works in the field of automatic face recognition and the number of publications has been constantly increasing in the last years (see Gross *et al.* [90] and Zhao and Chellappa [272] for literature surveys). The existing approaches can be roughly divided into two groups: *global* and *local* approaches of which the most important ones will be outlined in the following. Note that the majority of face recognition methods require the face to be localized in the input image beforehand. Some even necessitate a precise alignment of the face, such that it is in an upright position, centered and in a predefined scale.

Global Approaches

Global approaches process the face image as a whole mostly applying some type of statistical projection. The idea is to form a one-dimensional vector from the input image and project this (high-dimensional) vector into a sub-space of lower dimension where classification is supposed to be easier. Thus, such a projection – mostly linear – is supposed to select the most prominent features in order to discriminate the faces of different persons.

The most well-known of these approaches is the so-called *Eigenfaces* method introduced by Turk and Pentland [243]. Here, a PCA (*c.f.* section 2.2.1) is performed on the set of training face images projecting them into a linear subspace where the orthogonal eigen-vectors represent best the distribution of face vectors. The classification of new face images is then performed by projecting them into the linear subspace, forming a vector by selecting the eigen-values of the most discriminant dimensions and comparing these vectors to the ones of the training images. Much work has been done on the choice of the eigen-vectors. For example, Kirby and Sirovich [122] proposed a criterion based on a so-called *energy dimension* (see section 2.2.1) where the dimensions with the largest eigen-values are kept such that the normalized sum of these eigen-values is beyond a certain threshold (*e.g.* 90%). Martinez *et al.* [155] showed that the recognition rate can be improved by ignoring the first three dimensions, commonly deemed to account for illumination changes.

Another well-known approach presented by Belhumeur *et al.* [16] uses the Linear Discriminant Analysis (LDA) (*c.f.* section 2.2.2) and is called *Fisherfaces* as it maximizes the Fisher's criterion, *i.e.* a quotient of inter-class and intra-class variance. This means, it is not concerned with the best representation of the input data but rather with its maximum separability. However, since the dimension of the input vectors (the face images) is much greater than their number, there is a singularity problem when inverting the matrix of intra-class variance. To avoid this a PCA on the input vectors can be performed beforehand.

Numerous variants of these linear projection methods have been proposed in the literature [205, 236, 165, 233, 271]. Algorithms based on the Independent Component Analysis (ICA) have also been presented by Bartlett *et al.* [12].

More recently, Visani *et al.* [253] introduced an approach based on the Bilinear Discriminant Analysis (BDA) which realizes a two-dimensional linear projection of the lines and the columns of the face images. Then, they applied Normalized Radial Basis Function Networks to classify the projected vectors.

A technique called Discriminative Common Vectors (DCV) is due to Çevikalp *et al.* [36], which is a variation of the LDA for small sample sizes. For each individual they calculated a so-called common vector which is the result of a projection that eliminates features corresponding to the respective intra-class variation. Then, classification was performed using the common vectors.

There are also some approaches that do not use only *one* vector subspace but *several*. For example, the approach by Pentland *et al.* [181] computes a subspace for each orientation and scale of a face in the input image. A new face is then recognized by projecting the image into all subspaces and selecting the one where it is the closest to an vector of the face database.

Vasilescu *et al.* [247] have generalized this technique by using tensors. In their approach they proposed four-dimensional tensors corresponding to class, pose, illumination condition and facial expression respectively.

Other projection approaches [242, 262, 237, 261] construct a separate subspace for each class, *i.e.* each person.

A different face recognition approach has been introduced by Cootes *et al.* [41] called Active Appearance Models (AAM). As outlined in section 5.2, this method models independently shape and texture of the face by applying PCA respectively. The vector of shape and texture coefficients is then used for recognition. Thus, a new face that is to be recognized is fit to the model by an iterative optimization procedure and the resulting shape and texture parameters are compared to the ones from the training database. Lanitis *et al.* [131] applied this method for the first time to face recognition. Also, Edwards *et al.* [60] and Kumar *et al.* [128] presented face recognition methods based on AAMs.

Neural Networks have also been applied to automatic face recognition. For example, Lawrence *et al.* [132] proposed a method based on Self-Organizing Maps (SOM) to cluster the set of training images and thus construct a lower-dimensional subspace that conserves the topology of the input space. After projecting a face image onto the SOM, they applied a Convolutional Neural Network (CNN) in order to classify the signatures, *i.e.* the low-dimensional vectors.

Cottrell and Fleming [45] proposed to extract features by a non-linear projection using an Auto-Associative Neural Network (AANN). To classify the resulting features the applied a trained Multi-Layer Perceptron (MLP).

The approach of Lin *et al.* [146] uses probabilistic decision-based Neural Networks combining the advantages of statistical approaches and Neural Networks. Further, there are many approaches using Radial Basis Function Networks (RBFN) to classify feature vectors previously extracted, for example by a linear projection. Thomaz *et al.* [235] and Er *et al.* [62] proposed such methods.

Support Vector Machines (SVM), introduced by Vapnik [246], have also been used in the context of face recognition. Philips [184], for example, first used Bayesian subspaces to reduce the problem to two classes: inter-class and intra-class variations. Then, SVMs were applied for classification.

Finally, Jonsson *et al.* [119] presented a face authentication method using one SVM per person in order to classify the signatures obtained by PCA or by LDA.

Local Approaches

Local face recognition approaches rely on a separate processing of different regions in a face image. Mostly, this implies the extraction of local facial features. Often, local and global processing is combined in order to put the different local features into relation with each other. The most important of them are outlined in the following.

Brunelli and Poggio [33] proposed a technique that automatically extracts a set of 30 geometrical features from face images. In order to recognize a face, they compared these features pairwise using a Mahalanobis distance.

Another geometric approach has been proposed by Takács [234] who used binary contour maps extracted from the face images by a Sobel filter. The similarity between two contours was then computed by using a variant of the Hausdorff distance. This approach has been extended by Gao *et al.* [80] by transforming the contour maps into so-called Line Edge Maps (LEM) containing a list of line segments. The distance measure, however, was the same as the one Takács used.

Pentland *et al.* [181] introduced the approach of "Modular Eigenspaces" performing a PCA and classification on separate facial regions, *i.e.* the eyes, the nose and the overall face image. The mouth region showing much variation due to facial expressions leaded to a decrease of the recognition rate in their experiments.

The approach due to Heisele *et al.* [96] first detects the face region and 10 facial feature points using their previously published face and facial feature detector [97]. Then, the image regions around the respective facial features are extracted. Finally, the concatenated line (or row) vectors of these regions were combined to one large vector that is classified by a SVM.

Face regions were also extracted by a method proposed by Price and Gee [189]. Here, the authors considered three different image regions: a rectangular band going from the forehead to the bottom of the nose, one containing the two eyes and one being the entire image itself. Finally, they applied a variant of LDA to classify the respective face regions.

Another approach called Local Component Analysis (LCA) is due to Penev and Atick [180]. They performed several PCAs to extract different local features and combined them in a sort of deformable grid. A procedure that minimizes the reconstruction error finds then the optimal set of local grids.

Samaria *et al.* [214] introduced an approach based on Hidden Markov Models (HMM). Here, the face images were segmented into a certain number of overlapping sub-bands, and these sub-bands were in turn concatenated to a large one-dimensional vector or compressed by DCT. Then for each class, *i.e.* each person, a HMM was created reflecting the probabilistic sequence of sub-bands. New face images were classified applying the well-known Viterbi algorithm in order to compare the sequence of respective sub-bands with the trained models.

This approach has also been extended to so-called pseudo-2D HMMs [214, 169] which model horizontal and vertical sequences of image blocks. However, the blocks are not completely connected to each other, hence the approach is called pseudo-2D (see section 2.4.4).

Perronnin *et al.* [182] proposed an approach based on a 2D HMM framework where facial expressions and illumination variations were independently modeled. In this framework, a grid is placed on the face image where at each

location the system is assumed to be in some unknown state which depends on its adjacent states and is modeled by the transition probabilities of the HMM. This ensures the global consistency of the model. The local transformations, *i.e.* grid displacements and intensity variations, are represented by the states of the HMM. Moreover, at each position an observation is emitted according to the state-conditional emission probabilities which correspond to the cost of the local mapping. These mapping costs are modeled by multi-variate Gaussian mixture models. Finally, this probabilistic model of possible local transformations allows the establishment of a distance measure between a known face image and a unknown face image and thus face identification.

A different approach called Elastic Bunch Graph Matching (EBGM) has been presented by Wiskott *et al.* [260]. Here, faces are represented by so-called Face Bunch Graphs (FBG), where each node of the graph is associated with the possible appearances of a certain facial feature, *e.g.* the left or right eye. These appearances are represented by so-called *jets*, *i.e.* sets of 40 complex coefficients being the response of specific Gabor wavelet filters at a particular image position. The edges of the graph are labeled with the mean distance of the respective adjacent features. Once the FBGs have been created for each person using a training set of manually labeled face images, they can be applied to an unknown face by a specific matching algorithm. This algorithm iteratively tries to fit a graph to the face image by minimizing a similarity function taking into account both the geometrical similarity and the similarity of appearance of the features at each node. The final recognition is performed using another similarity criterion only based on the Gabor jets.

6.3 Face Recognition with Convolutional Neural Networks

6.3.1 Introduction

The drawback of most of the *global* approaches is their sensitivity to illumination changes. This problem is mainly due to the *linear* processing, whereas, under varying lighting conditions, the appearance of a face image undergoes a *non-linear* transformation. On the other hand, the drawback of *local* methods is that they often require an empirical choice of parameters, *e.g.* number of scales and orientations of Gabor filters or the positions where to apply the filters, which makes their implementation cumbersome and difficult.

We propose an approach [56, 82] that alleviates these problems by using a special type of Convolutional Neural Network (CNN) that learns to reconstruct from any face image of a given database a reference face image that is supposed to represent "best" the respective person and that is chosen beforehand. Figure 6.1 illustrates this in a basic schema with two example persons.

The "bottle-neck" architecture of the Neural Network actually learns a non-linear projection of the face images into a sub-space of lower dimension and then reconstructs the respective reference images from this compressed representation (*c.f.* section 2.8.4 on Auto-Associative Neural Networks). By using a CNN, an *empirical* choice of filter parameters is not necessary. Instead, the Neural Network learns these filters conjointly with the projection and reconstruction parameters while minimizing the overall reconstruction error. After training,

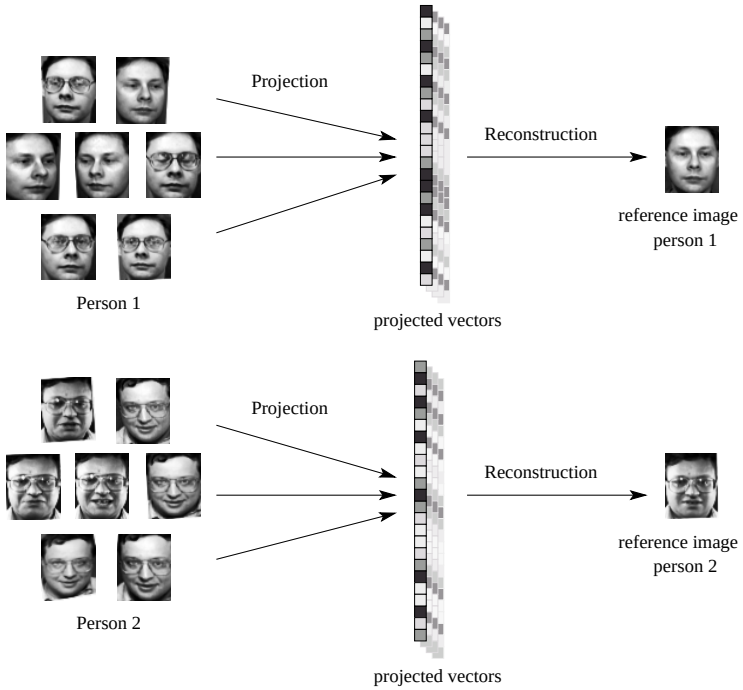


Figure 6.1: The basic schema of our approach showing two different individuals

face images can be classified by calculating distances between the respective *projected vectors* in the intermediate layer of the network or the distances between the respective *reconstructed images* at the output of the network.

6.3.2 Neural Network Architecture

The proposed neural architecture is a specific type of CNN consisting of six layers, where the first layer is the input layer, the three following layers are convolutional and sub-sampling layers, and the last two layers are standard feed-forward neuron layers. For details on CNNs, we refer to chapter 3 and in particular section 3.2.2. Our system is trained to transform an input face image into a reference image which is automatically chosen for each person to recognize. Figure 6.2 gives an overview of the architecture.

The retina l_1 receives a cropped face image of 46×56 pixels, containing gray values normalized between -1 and $+1$. The second layer l_2 consists of four feature maps which are all connected to the input map and which perform a convolution with a trainable 7×7 kernel followed by a linear activation function. The third layer l_3 sub-samples its input feature maps by a factor of two and uses a sigmoid activation function. Layer l_4 is another convolutional layer with 5×5 kernels. It contains three feature maps, each connected to two preceding maps as illustrated in figure 6.2. By combining the results of the low-level feature detectors, like edges or corners, it extracts higher-level features corresponding to more characteristic forms or patterns of a face image. Unlike the second layer, a sigmoid activation function is used here.

While the previous layers act principally as local feature extraction layers,

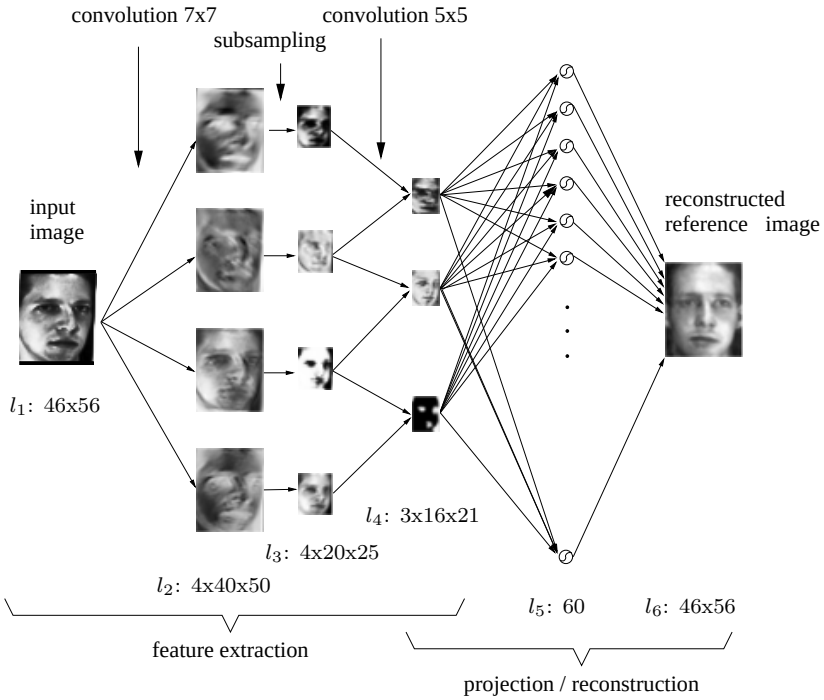


Figure 6.2: Architecture of the proposed Neural Network

layers l_5 and l_6 use the local information to form a global model. Layer l_5 is composed of a reduced number of neurons fully connected to layer l_4. This is the so-called "bottle-neck" of the network where a compact representation of the input face images is learned.

The architecture of this part of the network is inspired by *Auto-Associative* Neural Networks (AANN) (see section 2.8.4) which are trained to reproduce an input pattern at their outputs while using a hidden layer containing much fewer neurons (bottle-neck) than the input and output layers. Here, a so-called hetero-association is performed in the last three layers, because the desired output in layer l_6 is different from the output of layer l_4. Moreover, the activation functions of the neurons in l_5 are non-linear leading to a non-linear projection and dimensionality reduction.

In our proposed architecture, Layer l_5 contains 60 neurons with sigmoid activation function and the output layer l_6 is composed of an array of neurons of size 46×56 representing a gray-scale image normalized between -1 and $+1$. These neurons are fully connected to the preceding neurons and use a linear activation function.

6.3.3 Training Procedure

The Neural Network is trained using a face database with a fixed number N of individuals. For each individual, several images with varying pose, illumination and facial expressions are necessary. Before the actual training the face database is divided into a training and a test set as will be explained in section 6.3.5.

The training procedure consists of two successive steps:

1. selection of the reference images and
2. actual training of the Neural Network.

These steps are detailed in the following.

Choosing the Reference Images

Let us denote im_{ij} the j-th example image of individual $i = 1..N$ in the face database ($j = 1..M_i$). For each individual i a reference image r_i among the face images im_{ij} in the training set has to be chosen. The Neural Network is then trained to respond for any input image im_{ij} of a given individual with the respective reference image r_i for that individual. In this way, it will learn to extract features invariant to the intra-class variations present in the training images, *e.g.* pose, illumination or facial expressions.

We experimented with two different strategies for choosing the reference images. They are both based on a Euclidean distance measure between the respective image vectors im^+_{ij}, defined as the one-dimensional vectors obtained by concatenating the rows of the respective images im_{ij}. The strategies we investigated are the following:

1. *Choose the most representative image:*
   The face image of the individual i that is closest to the mean image $\overline{im^+_i}$ of i is chosen:
   $$r_i = \underset{im^+_{ij}}{\operatorname{argmin}} \|im^+_{ij} - \overline{im^+_i}\| \;\; \forall\, i \in 1..N,\; j \in 1..M_i\;. \tag{6.1}$$

2. *Choose the most distant image:*
   The face image of the individual i that has the greatest distance to the face images of all the other individuals is chosen.
   $$r_i = \underset{im^+_{ij}}{\operatorname{argmax}} \|im^+_{ij} - im^+_{kl}\| \qquad \forall\, i,k \in 1..N,\; j \in 1..M_i,\; l \in 1..M_k,\; k \neq i\;. \tag{6.2}$$

We will call these strategies MEAN and DIST in the following.

Training the Neural Network

In order to construct the training set for the Neural Network the face images are normalized in the following way. First, each image is cropped in such a way that the face is centered and that the eyes and the mouth are roughly at predefined positions while keeping the aspect ratio. Then, each image is histogram-equalized and resized to the dimensions of the retina l_1 (46×56). Training is performed using the Backpropagation algorithm which has been slightly adapted to account for the shared weights in layers l_2 to l_4 (see Alg. 8 on page 57). For a given example im_{ij} the objective function is the following:

$$E_i = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} \left(d_i(x,y) - y_6(x,y)\right)^2 \quad , \tag{6.3}$$

which is the mean squared error (MSE) between the computed outputs $y_6(x, y)$ for training example im_{ij} and the desired outputs $d_i(x, y)$, where d_i represents the respective reference image r_i normalized between -1 and $+1$. Before the actual training, the weights are initialized at random. Then, they are updated after each presentation of a training example (online training). Training is stopped after 8000 iterations.

Note that by training the Neural Network, *i.e.* by minimizing the objective function, all parameters are learned conjointly: the convolution filters, the projection and the reconstruction parameters. In other words, the proposed architecture optimizes the filters and, at the same time, the projection parameters in order to reconstruct best the respective reference images. This is a clear advantage compared to most other projection methods where separate pre-processing and projection steps necessitate a "manual" integration and parameter determination.

6.3.4 Recognizing Faces

Once the Neural Network is trained with a certain number of individuals, it can be applied to new, unknown face images of the same individuals in order to recognize them. To this end, a given face image is cropped and normalized in the same way as the training images (cf. section 6.3.3) and presented to the Neural Network. The Neural Network then tries to reconstruct the reference image corresponding to the respective individual. Finally, a simple nearest neighbor classification based on the Euclidean distance between the Neural Network's output and all the reference images identifies the individual shown on the input face image.

More formally,

$$I = \underset{i}{\operatorname{argmin}} \|y_6 - d_i\| \quad \forall\, i \in 1..N \;, \tag{6.4}$$

where I is the resulting identity, y_6 is the output of the Neural Network and d_i is the reference image of individual i, normalized between -1 and $+1$.

In our experiments, however, we slightly modified this classification algorithm for efficiency reasons. Instead of classifying the outputs of the final layer l_6 we used the outputs of the neuron layer l_5 which represent the projected vectors. We then compare the projected vectors with the ones produced by the reference images. Thus, the classification formula becomes:

$$I = \underset{i}{\operatorname{argmin}} \|y_5 - v_i\| \quad \forall\, i \in 1..N \;, \tag{6.5}$$

where v_i represents the the output of layer l_5 when presenting the reference image r_i of the i-th person to the Neural Network.

The two classification formulas led to equivalent results but the second one is more efficient in terms of computational time. Thus, all the results presented in this paper were obtained using Eq. 6.5.

6.3.5 Experimental Results

We conducted experiments on two public face databases: the Olivetti Research Ltd. (ORL) face database [214] and the Yale database [16] (see Appendix A).

The ORL database contains 40 individuals with 10 images per individual showing slight pose variations, facial expressions and rather limited illumination changes. The Yale database contains only 15 individuals with 11 images

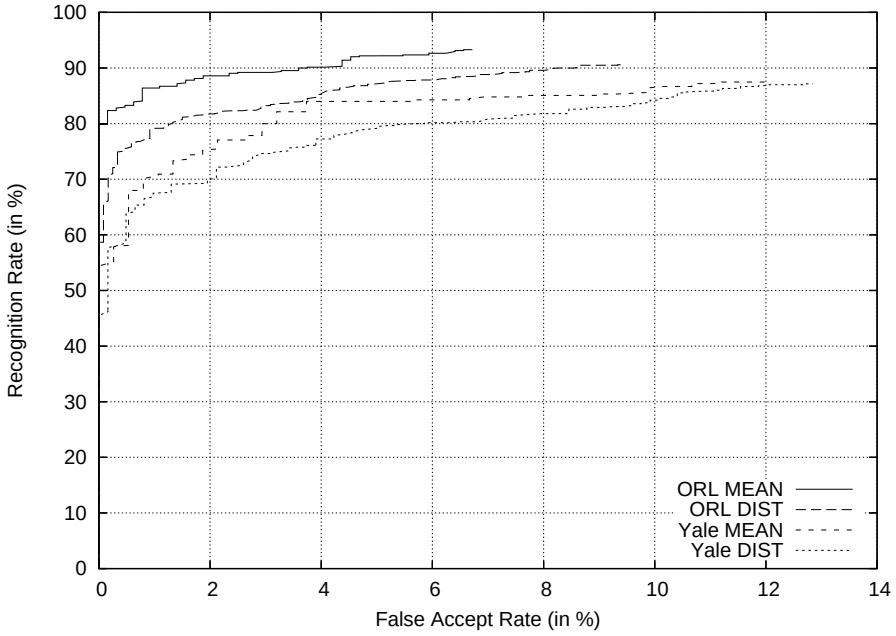


Figure 6.3: ROC curves for the ORL and Yale databases

each. They show virtually no pose variations but much more illumination variations (*e.g.* left/right/center light) and facial expressions (smile, sad expression, open/closed mouth).

In order to assess the different approaches we performed a "leave-one-out" validation. Thus, the Neural Network was initialized and retrained 30 times with a random separation into training and test set, where one example per class is used for testing and the rest for training. Then, the mean of the final recognition rates on the respective test sets was calculated.

Figure 6.3 shows the Receiver Operator Characteristic (ROC) curves of the proposed approach with both reference image selection strategies, MEAN and DIST. The ROC curves illustrate the recognition rates *vs*. the false accept rate while varying a distance threshold above which a face image is rejected. In general, the recognition rate of the ORL database is higher than that of the Yale database. The recognition rates without rejection are shown in table 6.1. Further, the MEAN approach performs better than the DIST approach (*c.f*. section 6.3.3) for both test sets. Thus, for the following experiments only the results using the MEAN strategy are presented. Figure 6.4 illustrates some input face images (top row), the reconstructions (middle row) and the respective reference images (bottom row).

We also compared the proposed approach with the Eigenfaces and the Fisherfaces approaches. In our implementation of the Eigenfaces method we performed a projection onto the most significant eigenvectors such that the corresponding eigenvalues represent 90% of the total energy, that is 73 eigenvectors for the ORL database and 35 eigenvectors for the Yale database. The first three directions, however, were not used for classification as they are considered to account mainly for lighting variations [16]. As for the preceding experiment, a leave-one-

(a) ORL (b) Yale

Figure 6.4: Examples of image reconstruction. *Top row:* input images, *middle row:* reconstructed images, *bottom row:* reference images

	ORL	Yale
Eigenfaces	89.7%	77.9%
Fisherfaces	87.7%	85.2%
proposed approach: DIST	90.6%	87.1%
proposed approach: MEAN	**92.6%**	**93.3%**

Table 6.1: Overview of the recognition rates

out validation with the same training and test sets was performed. Note that concerning the classification procedure the proposed approach is more efficient in terms of computation time and memory usage because it only requires the reference images in order to classify new face image whereas the other two approaches need the whole dataset. Figures 6.5 and 6.6 show the ROC curves of the Eigenfaces and Fisherfaces methods together with the proposed approach for the ORL and Yale database respectively. For both databases the proposed method clearly outperforms the other methods. Table 6.1 summarizes the recognition rates of the preceding experiments.

We further evaluated the robustness of our approach with respect to noise and partial occlusions. In the first experiment, we added Gaussian noise with increasing standard deviation σ to the individual pixels of the images of the test set. Figure 6.7 shows the respective recognition rates with varying σ.

Note that a σ of 0.5 represents a considerable amount of noise as the gray values are between -1 and $+1$ (see illustration at the bottom of Fig. 4.13). The graphs show that the proposed method is very robust to Gaussian noise. For $\sigma < 0.5$ the recognition rate decreases by only 12% for the ORL database and by only 6% for the Yale database, and it remains above 80% for $\sigma < 0.6$. The Eigenfaces approach showed a slightly better performance in this particular experiment, the recognition rate staying almost constant over the whole interval. This can be explained by the pure global processing of the PCA. As it is extracting rather lower frequency features it is less sensitive to high-frequency noise.

The last experiment demonstrates the robustness of the approach with respect to partial occlusion. To this end, the bottom part of the images is masked by a black band of varying height. Figure 4.14 shows the respective results

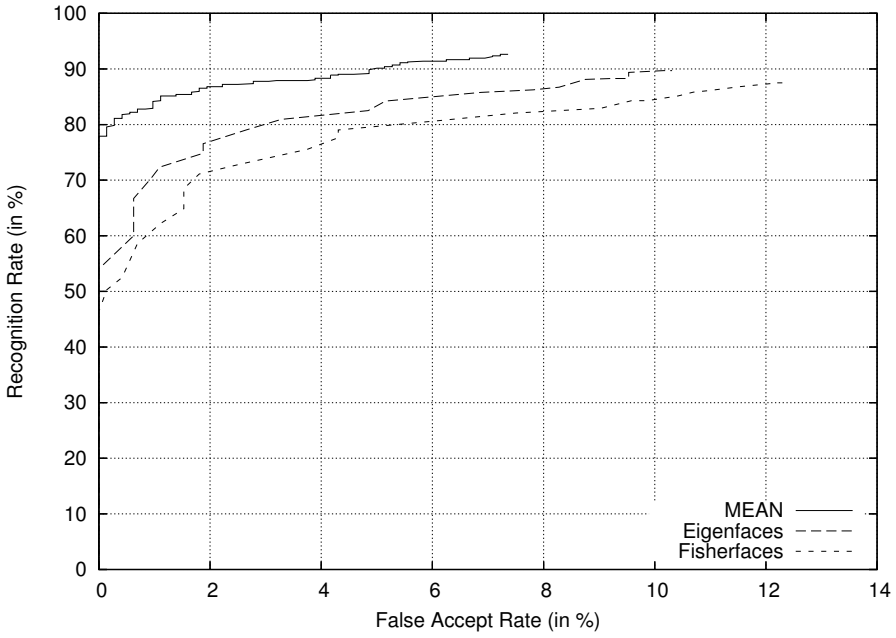


Figure 6.5: Comparison with the Eigenfaces and Fisherfaces approach: ORL database

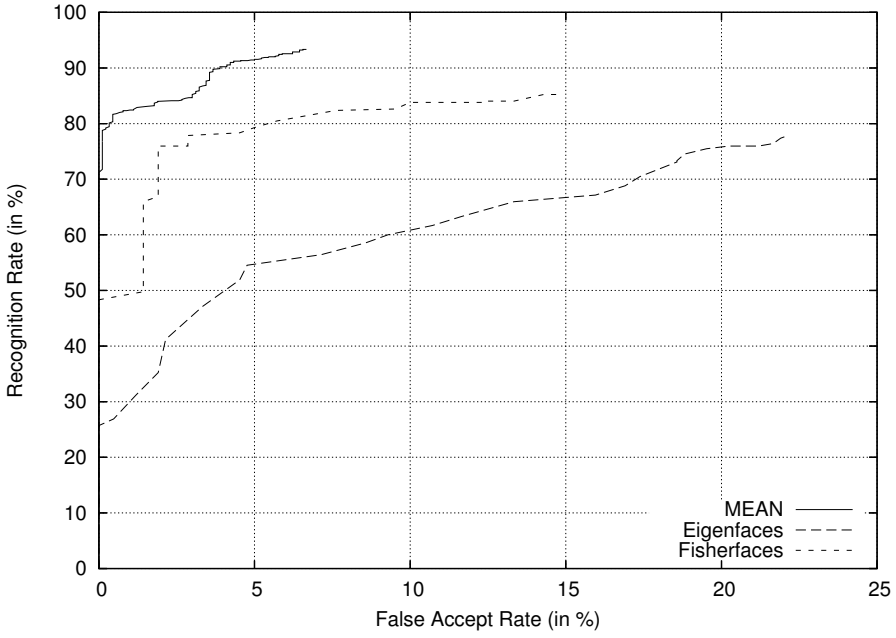


Figure 6.6: Comparison with the Eigenfaces and Fisherfaces approach: Yale database

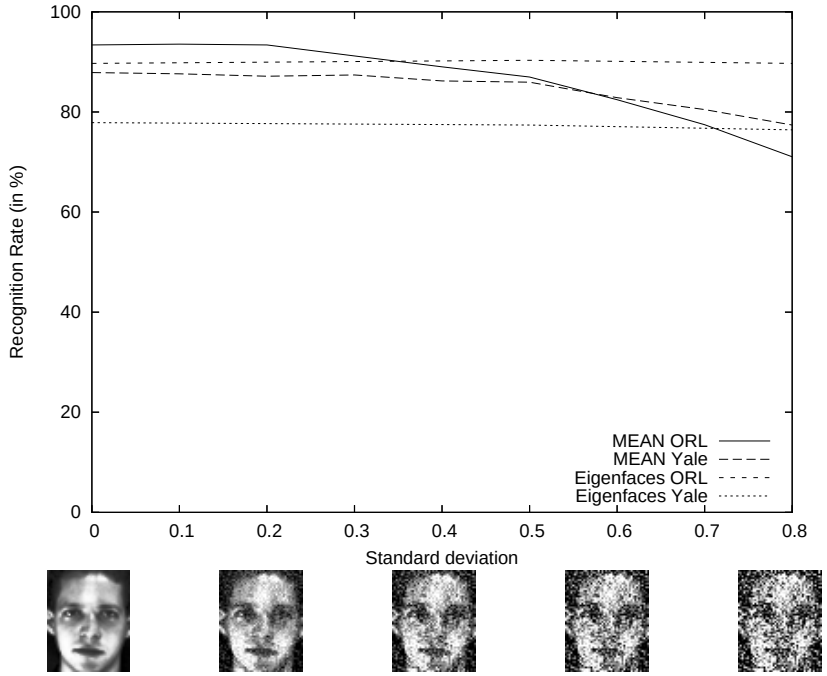


Figure 6.7: Sensitivity analysis of the proposed face recognition approach: Gaussian noise

as well as some example images at the bottom to illustrate the type of occlusion (0%, 10%, 20%, 30% and 40%). Here, our approach clearly outperforms the Eigenfaces method. For both databases the recognition rate stays above 80% when the occluded proportion is less than 20% of the image, whereas the performance of the Eigenfaces method drops considerably.

6.4 Gender Recognition

6.4.1 Introduction

Automatic gender recognition, *gender classification*, or sometimes *sex classification* of faces, denotes the problem of recognizing the gender, *i.e.* male or female, of a person using a face image. Hence, it is a binary classification problem which, in principle, can be tackled by any discriminative or generative classification approach, *e.g.* a Bayes classifier, Neural Networks, Support Vector Machines. There are many applications for automatic gender classification using face images, *e.g.* image and video indexation, or in the context of person identification, or systems (TV, home computer) that automatically adapt the presented information to the user sitting in front of a camera.

However, the problem of gender classification by face images is far from being trivial as the distinctive features between faces of men and women can be rather subtle. In some cases, even humans are not capable of telling the gender of a person only by one photograph showing the face of the person. In fact, when

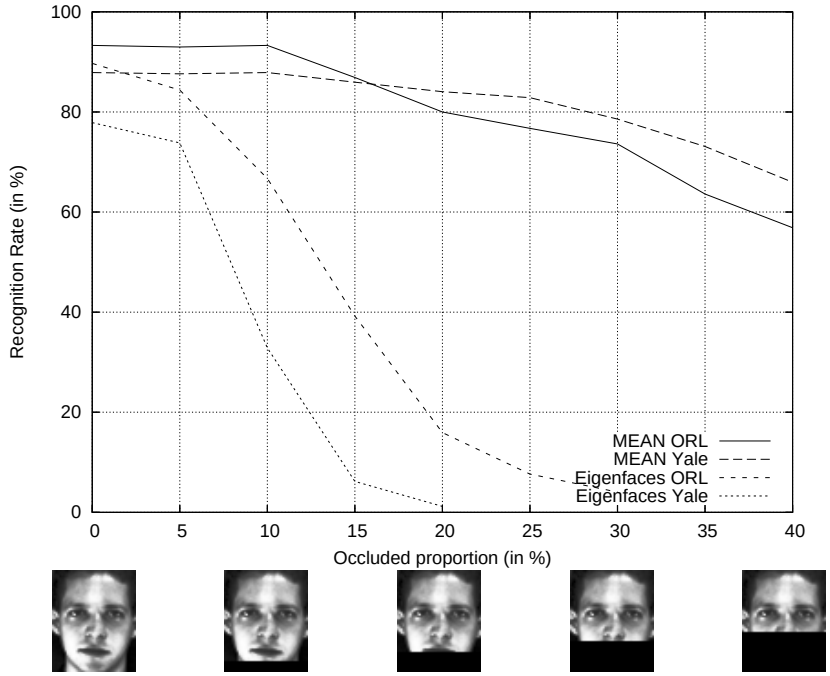


Figure 6.8: Sensitivity analysis of the proposed face recognition approach: partial occlusion

meeting an unknown person, human beings often take into account other aspects such as hair, voice, gait etc. This suggests that the decision boundary of the two classes is non-linear and rather complex, and for this reason machine learning techniques appear to be an appropriate approach as the resulting classifiers show to be capable of extracting discriminant features from the high-dimensional input space, *i.e.* the face images.

In this chapter, we will briefly outline some existing methods for gender recognition and then propose a CNN-based approach and present some experimental results.

6.4.2 State-of-the-art

One of the first works on automatic gender recognition was the approach proposed by Golomb *et al.* [86] called SEXNET which was based on a MLP and achieved an accuracy of 91.9% on a database of 90 images.

Other early works use geometric features instead of image intensity. Brunelli and Poggio [32], for example, classified 16 point-to-point distances with two competing RBF Networks, for male and female examples. They used a training database of 168 images and reported a 79% accuracy on novel faces.

Fellous [66] also used geometric features, in this case 22 horizontal and vertical point-to-point distances. With a subsequent statistical analysis using PCA, he retained the most discriminative features for gender classification and obtained a 90% accuracy.

A graph-based approach has been presented by Wiskott *et al.* [259]. The

nodes of the graph represent wavelet-based local appearances and the edges are labeled with distance vectors. Using a small number of model graphs for male and female face images they were able to classify new images by a special graph matching procedure with a 91.2% classification accuracy.

O'Toole *et al.* [177] used a PCA-compressed representation of the image intensities and classified them with a linear Perceptron.

PCA has also been used by Graf *et al.* [89] who compared it to Local Linear Embedding (LLE), a non-linear neighborhood-preserving dimensionality reduction technique. The authors then employ SVMs for the final gender classification. Sun *et al.* [230] additionally selected a subset of PCA features by a Genetic Algorithm-based method and then compare different classification approaches, *i.e.* Bayes classifier, a LDA-based classifier, a Neural Network and a SVM, on the gender recognition problem. BenAbdelKader and Griffin [18] also applied PCA on the pixel intensities. For classification they compare a SVM approach with Fisher's Linear Discriminant.

A method using ICA for feature extraction has been proposed by Jain and Huang [113]. The final classification was performed by means of a LDA.

Buchala *et al.* [34] used a non-linear dimensionality reduction method called Curvilinear Component Analysis (CCA) and then compared SVMs with MLPs for gender classification.

Moghaddam and Yang [164] proposed an approach training an SVM classifier directly on the image pixel intensities. A correct gender classification rate of 96.6% on the FERET database has been reported by the authors.

A different method is due to Gutta *et al.* [92]. Here, a mixture of experts combines several classifiers, RBF ensembles, Inductive Decision Trees (IDT) and SVMs which were employed on different image regions. The authors reported a classification rate of 96% on the FERET database.

Lanitis *et al.* [130] presented an approach that transforms the face images into a shape-free patches using AAMs. The resulting images were then classified using a Mahalanobis-based distance measure. AAMs have also been used by Saatci *et al.* [209] for feature extraction. Here, gender recognition was performed by matching a trained AAM to the unknown face and finally classifying the resulting model parameters by a SVM. A classification of 97.6% has been obtained on a set of several databases but the results only contain the cases where the AAM converged correctly.

Shakhnarovich *et al.* [222] presented an approach based on Adaboost. They proposed to use the features from the Boosted Cascade Detector for face detection from Viola and Jones [250] to perform gender classification and report an accuracy of 79%. Recently, Baluja *et al.* [11] also proposed an Adaboost-based method. Here, they used features comparing pixel intensities for their weak classifiers in the Adaboost procedure and compared it to a SVM-based approach. A maximum accuracy of over 93% has been achieved by their gender recognition method on the FERET database.

Finally, the approach presented by Tivive and Bouzerdoum [241] uses Shunting Inhibitory Convolutional Neural Networks (SICoNNets) (*c.f.* section 3.4.4). They obtained an 97.1% accuracy on the FERET database.

Figure 6.9: Examples of training images for gender classification. *Left:* men, *right:* women

6.4.3 Gender Recognition with Convolutional Neural Networks

We propose a gender recognition method [82] based on a Convolutional Neural Network (CNN) architecture similar to the one of the Convolutional Face Finder (CFF) from Garcia and Delakis [81] described in section 4.2.3. The only significant change compared to the CFF architecture is the size of the feature maps. The dimension of the retina of the gender recognition CNN is 46×56 pixels. The size of feature maps of all of the other layers follows systematically from the retina dimension and the respective kernel sizes, *i.e.* 5×5 kernels for the first and a 3×3 kernels for the second convolution layer. The sub-sampling window size is 2×2 as with the CFF. In the final layer there is a single neuron representing the classification result, *i.e.* the value -1 for male and $+1$ for female face images.

The training set was constructed using cropped and histogram-equalized frontal face images of the FERET database [183] (see Appendix A), where 80% of the images were used for training and 20% for testing. Figure 6.9 shows some training examples situated close to the border of the decision boundary. Training was then performed using the online Backpropagation algorithm for CNNs (see Alg. 8 on page 57).

The classification rate obtained with this architecture is 94.7% which is comparable to the results obtained by [11] who reported a (maximal) classification rate of 93.5% using a SVM-based and 94.4% using a Adaboost-based approach. Figure 6.10 shows the respective ROC curve of the test set.

Note that many existing gender classification methods mix the same persons across the training and test set when there are several examples of one person's face, which obviously makes the problem easier as the algorithm might "memorize" specific individuals. In the preceding experiment, we put *all* example images of one particular person in *either* the training *or* the test test, demonstrating more soundly the generalization capacity of the system.

We also conducted an experiment with a mixed training set, *i.e.* the test images were randomly selected from the whole database (FERET) such that example images of one person might be in the training *and* the test set. As expected, the recognition rate is higher than with the unmixed datasets: 97.4% of correct classification have been obtained with this setting, which is also comparable to the results obtained by [11] where the best SVM achieves an accuracy of 97.1% and the best Adaboost classifier: 96.6%.

6.5 Conclusion

In this chapter, we presented a face recognition approach based on a specific type of CNN which is trained to reconstruct reference images pre-defined for

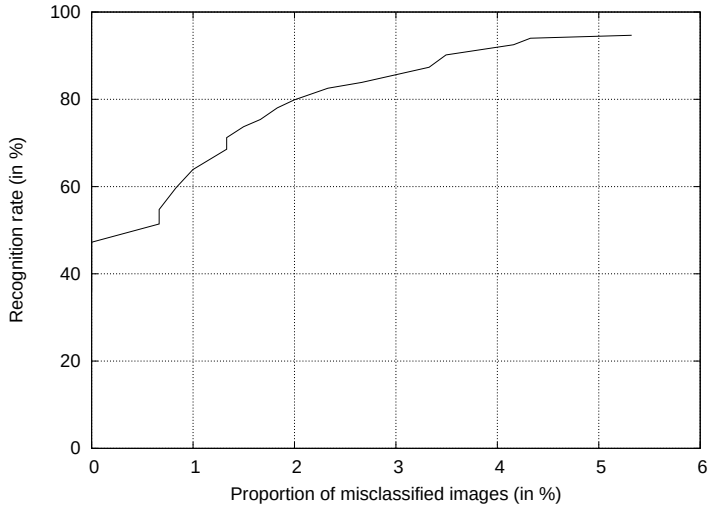


Figure 6.10: ROC curve of the gender recognition CNN applied to the unmixed FERET test set

each person of the training set to identify. By reconstructing images the system actually learns a non-linear mapping onto a low-dimensional sub-space which is robust to the intra-class variations present in the training set, *e.g.* illumination and pose variation or facial expressions. The advantage of this approach is the tight integration of local and global processing in one neural architecture and the ability to learn all parameters conjointly without requiring a manual choice of these parameters, *e.g.* the filter masks or the projection directions.

We trained this system on several public databases and showed its superior performance in terms of recognition rate w.r.t. classical face recognition methods. Further, we experimentally showed its robustness to external influences by manually adding a considerable amount of pixel noise and partial occlusions.

Finally, we also presented a solution for the gender recognition problem of faces. We employed a CNN with a architecture similar to the one of the CFF and trained it on images of the FERET database. We obtained 94.7% of correct gender classification which is comparable to the results of Adaboost- and SVM-based methods previously published in the literature.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion

In this work, we described the problem of automatic appearance-based facial analysis and outlined some of the most common techniques to tackle it. We particularly focused on an approach called Convolutional Neural Network (CNN) which is a connectionist model inspired by biological findings in the visual system of mammals. Being initially used for handwritten digit recognition it has meanwhile been successfully applied to many other visual pattern recognition problems. For example, the CNN-based face detection system proposed by Garcia and Delakis [81] shows to be very robust against most types of noise and external influences that face images can undergo in real-world settings, and the results are superior to other state-of-the-art appearance-based classification methods evaluated on common public databases.

Using CNN models that are, in a way, similar to the ones mentioned above, we proposed effective approaches to different facial analysis problems. The first system we presented tackles the problem of *face alignment* which is a crucial step in many facial image processing applications, notably face recognition. This procedure is typically performed *after* face detection and *before* face recognition, and its purpose is to align the bounding rectangles coming from the face detection system such that specific facial features are roughly at predefined positions inside the images bounded by the respective rectangles. The proposed approach is based on a specific CNN architecture that learns the parameters of this type of affine transformation, *i.e.* x/y translation, rotation angle and scale factor, by means of annotated mis-aligned face images. The trained system is then able to conjointly estimate these four transformation parameters when presenting a mis-aligned face image at its input without explicitely localizing specific facial features such as the eyes. We experimentally showed that in an iterative approach, it is able to precisely and robustly correct the bounding rectangles coming from a face detector such that they are well-aligned with the faces inside the respective rectangles. A correct alignment rate of 94% for our own test set containing difficult images and 80% for the public BioID database has been obtained while tolerating a mean distance from the desired bounding

rectangle corners of 10% of the rectangle's width.

Then, we presented a central problem in facial analysis, called *facial feature detection*, *i.e.* the localization of characteristic feature points in face images. We further proposed a CNN-based approach to this problem which, in a hierarchical manner, localizes 10 facial feature points in face images taken under difficult conditions. The system correctly localizes the 10 points of 96% of the faces of the AR database and 87% of the BioID database tolerating an error of 10% of the inter-ocular distance. We also measured the precision of eye pupil detection for the BioID database and showed that our approach outperforms state-of-the-art systems. Different types of input features to the CNN have been tried, while normalized image intensities gave the best results compared to gradients and Gabor wavelet filter responses. Finally, we demonstrated the considerably high robustness of the proposed approach w.r.t. Gaussian pixel noise and partial occlusions.

In the subsequent chapter, we described the problems of automatic face recognition and gender classification. For the former task, we proposed a novel approach based on a specific CNN architecture which learns a non-linear mapping from the image space onto a lower-dimensional sub-space. By learning to reconstruct reference images for each person to classify the system builds a sub-space less sensitive to the intra-class variations present in the training set. The recognition rate obtained in our experiments on the public databases ORL and Yale is 93%, which is higher than the one obtained with the classical face recognition methods: Eigenfaces and Fisherfaces. As with the other proposed approaches, we also showed the robustness of the system w.r.t. Gaussian pixel noise and partial occlusions. Finally, we presented a gender classification method based on a model similar to the CNN used in the face detection system of Garcia and Delakis [81]. The system is able to correctly recognize the gender of 94.7% of the face images of a test set extracted from the FERET database when images of the same individuals are not mixed across the training and test set. This result is comparable to recognition rates of state-of-the-art systems based on SVMs or Adaboost.

To sum up, one can say that the CNN approach is a very powerful machine learning technique capable of learning very complex non-linear functions related to appearance-based pattern recognition. The advantage of CNNs is that they automatically learn adapted filters that extract non-linear characteristic features useful for a dedicated classification task. This integrated learning procedure of all inherent parameters using the Backpropagation paradigm and a global error function to minimize circumvents the somewhat tedious task of "manual" parameter determination of feature extractors and classifiers. Additionally, the possibility to easily implement CNNs on embedded and parallel platforms and to build dedicated real-time systems is a big advantage of this approach.

However, the drawback of this technique is the "manual" determination of the *topology* of the CNNs for a given task as well as the creation of a *sufficiently large and appropriate* training database. This leads us to the final section of this work which addresses some perspectives of future research in this field.

7.2 Perspectives

7.2.1 Convolutional Neural Networks

As mentioned in the previous section, the architecture of a CNN best suited for a given task is determined empirically, which is undoubtedly a *sub-optimal* choice. It would however be preferable to automatically adapt the complexity of the model to the complexity of the problem in order to reduce the size of the CNN and increase its generalization capacity. Many works have been conducted in this area, *e.g.* with respect to constructive neural networks, growing and pruning techniques or incremental learning. However most of them are concerned with standard neural architectures, *e.g.* MLPs, and do not cope with more complicated models such as CNNs which have specific types of architectures employing weight sharing.

Another interesting issue would be to investigate the ability of a CNN to learn from a *reduced* set of training examples while obtaining equal or almost equal results than with a *larger* training set. This implies either a technique to automatically select from a large set the most significant training examples for a given task or – if a larger training set is not available – a method to construct a type of model on the reduced training set and to integrate this model into the training procedure.

Finally, CNNs which are able to extract features *invariant* to larger translations, rotations and changes in scale would be an interesting area of future research. The works by Ranzato *et al.* [193] on shift-invariant feature extraction using a CNN-based encoder-decoder framework (*c.f.* section 3.4.5) are particularly promising in this regard. In this context, another interesting perspective could be pursued: the application of CNNs to whole scenery images in order to extract prominent feature points helping to classify these images or particular visual objects. This type of CNN could then be considered as a specialized and learned interest point detector and/or descriptor.

7.2.2 Facial analysis with Convolutional Neural Networks

Many face analysis problems, like face recognition under unconstrained real-world conditions, are still far from being resolved. Variations in illumination, head pose and facial expressions represent the biggest difficulties for most of the current solutions including CNNs. In this work, we haven't explicitely modeled these variations but learned them implicitly using training images taken under different conditions. An interesting direction of future research would thus be to integrate this type of model into the proposed CNN-based systems and investigate the its impact on the overall performance. An initial approach could be to train a CNN to estimate or even to correct the illumination of a face using annotated training examples. One could also imagine a CNN-based head pose estimator, the results of which could then be used for further view-specific processing, *e.g.* face recognition.

Finally, CNNs are not limited to gray scales but can also process color images, *i.e.* by simply presenting three images corresponding to the different color channels (*e.g.* red, green and blue) at the input. However, as experimental results on face detection show, the use of color might have a negative impact on the robustness of the system especially with respect to different lighting condi-

tions. Nevertheless, the ability of CNNs to process *multi-dimensional* data is a very interesting perspective as they could for example be applied to many types of multi-modal data, *e.g.* images and sound, images of different views, or video frames at different time steps etc. One could also imagine to process 3D data by CNNs, or to fuse 2D and 3D data, *e.g.* for face recognition.

Many interesting research directions in the domain of automatic facial analysis are to be explored. Convolutional Neural Networks represent a promising approach from the computational side but perhaps also novel biological and/or neuro-scientific results will yield further ideas to improve this model.

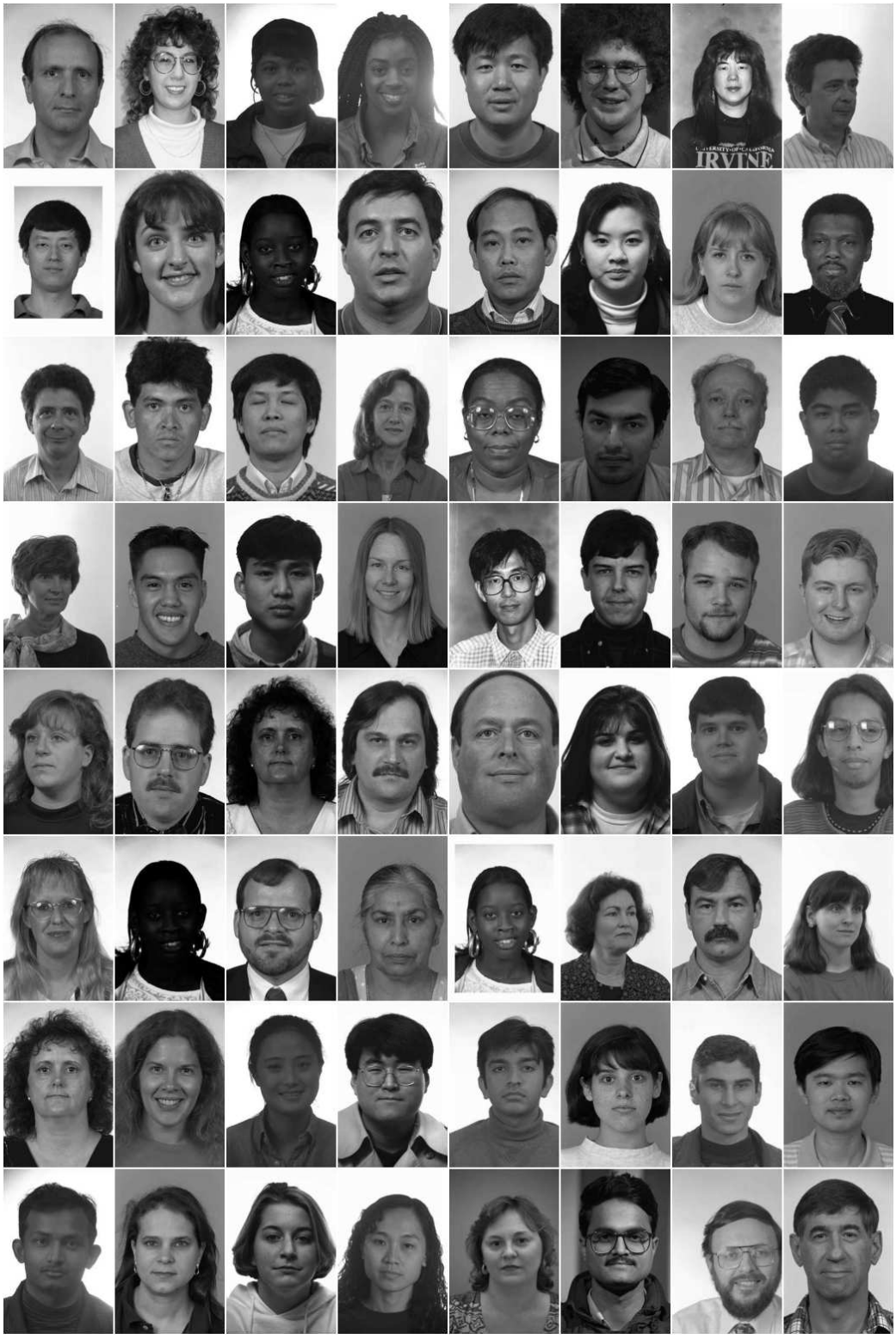
Appendix A

Excerpts from the used face databases

A.1 AR

A.2 BioID

A.3 FERET

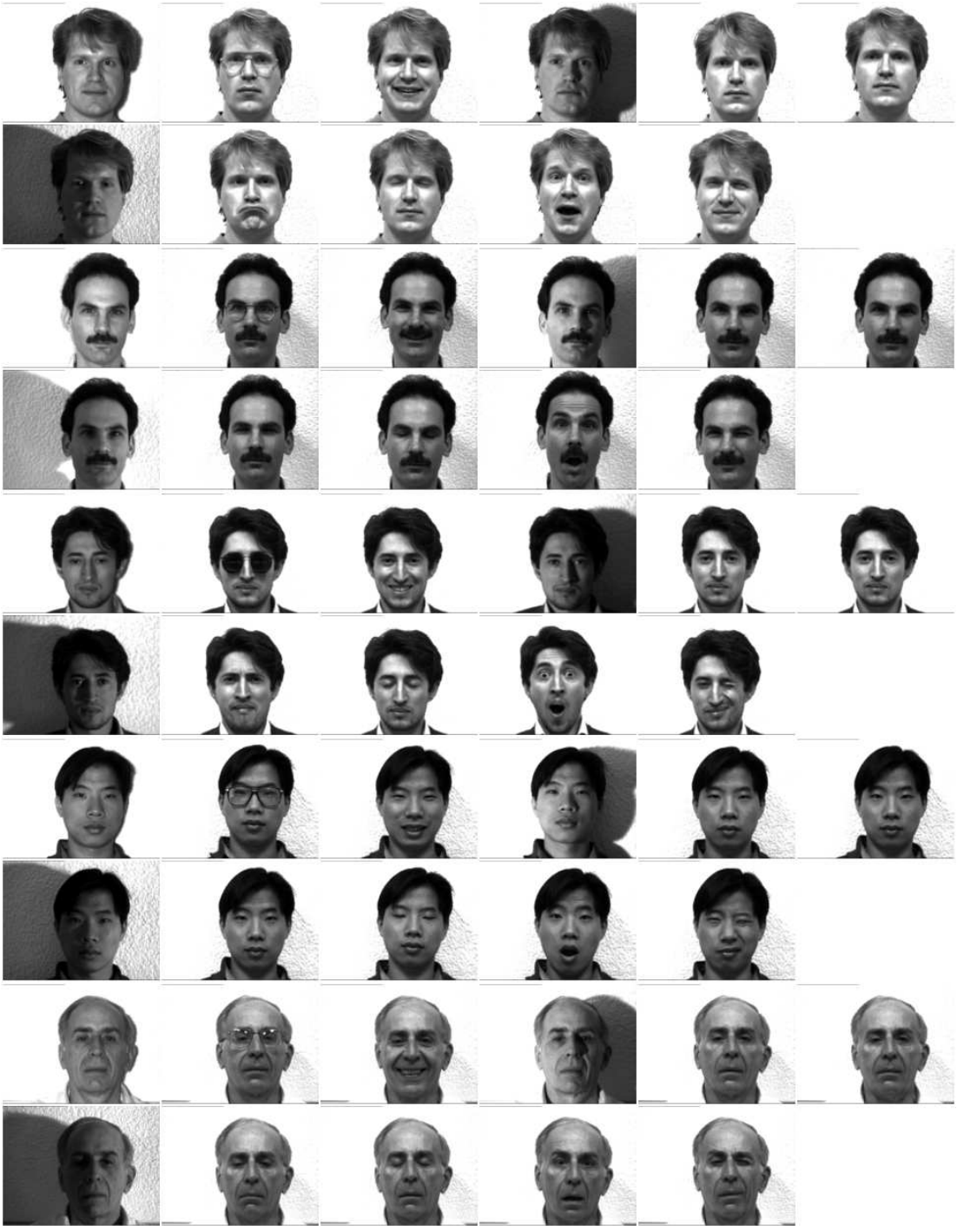


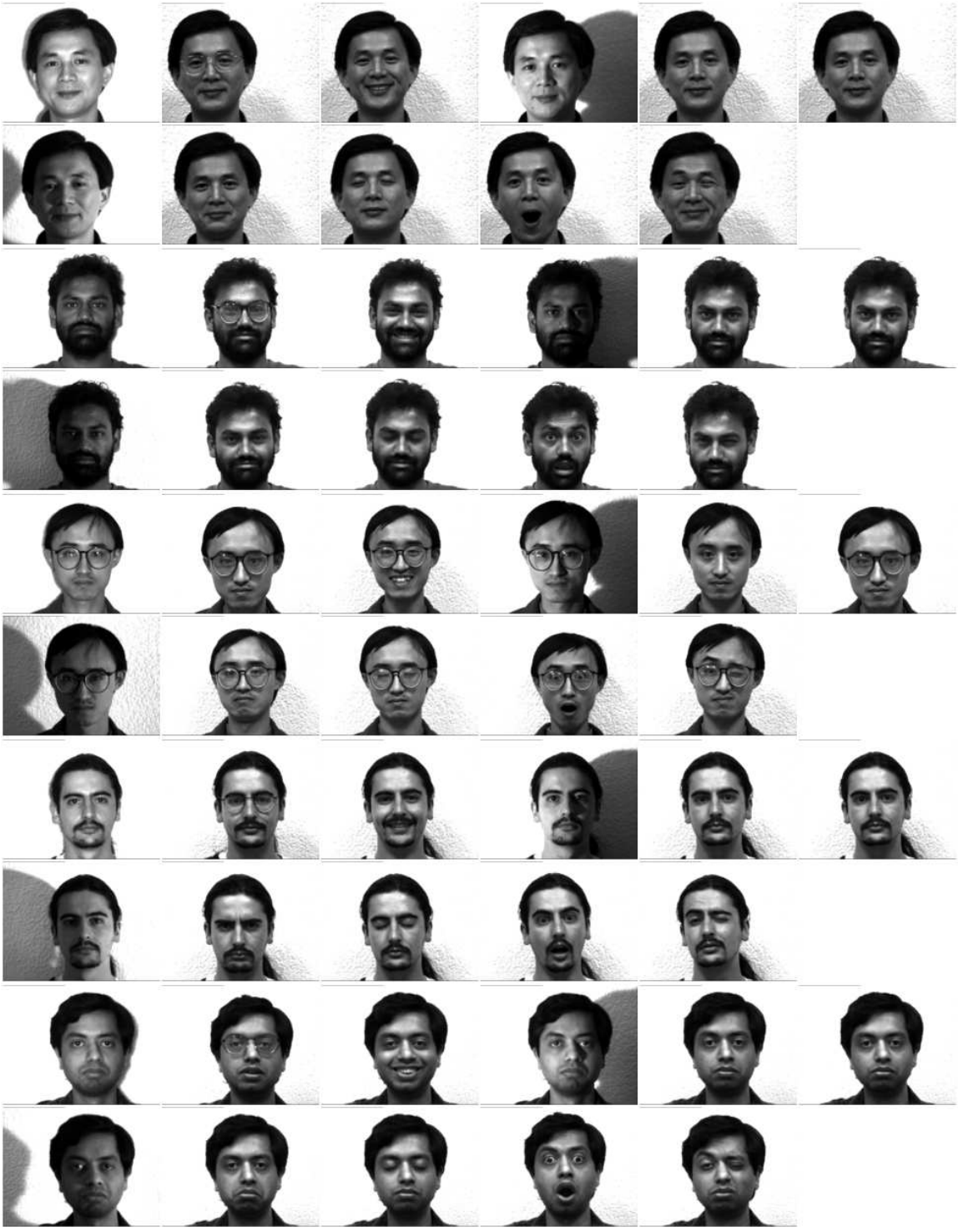
A.4 Google Images

A.5 ORL

A.6 PIE

A.7 Yale





Bibliography

[1] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. Parameter adaptation in stochastic optimization. In D. Saad, editor, *On-Line Learning in Neural Networks*, chapter 6. Cambridge University Press, 1999.

[2] S.-I. Amari, A. Cichocki, and H. Yang. A new learning algorithm for blind source separation. In D. S. Touretzky, M. C. Moser, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 757–763. MIT Press, Cambridge, MA, 1996.

[3] G. Arulampalam and A. Bouzerdoum. A generalized feedforward neural network architecture for classification and regression. *Neural Networks*, 16:561–568, 2003.

[4] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):602–605, June 1993.

[5] S. O. Ba and J. M. Odobez. A probabilistic framework for joint head tracking and pose estimation. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 264–267, August 2004.

[6] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[7] E. Bailly-Baillière, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariéthoz, J. Matas, K. Messer, V. Popovici, F. Porée, B. Ruiz, and J.-P. Thiran. The BANCA database and evaluation protocol. In *Proceedings of the Fourth International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 625–638, 2003.

[8] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Computer Vision and Pattern Recognition*, volume 1, pages 1090–1097, 2001.

[9] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *IEEE Transactions on Neural Networks*, 2:53–58, 1988.

[10] S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Neural Information Processing Systems*, 1998.

[11] S. Baluja and H. A. Rowley. Boosting sex identification performance. *International Journal of Computer Vision*, 71(1):111–119, 2007.

[12] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, 2002.

[13] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *International Conference on Computer Vision*, volume 2, pages 383–390, 2001.

[14] R. Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex Systems*, 3:331–342, 1989.

[15] P. Belhumeur, J. Hespanha, and D. Kriegman. What is the set of images of an object under all possible lighting conditions. *International Journal of Computer Vision*, 28(3):245–260, 1998.

[16] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegmann. Eigenfaces vs Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):711–720, 1997.

[17] A. Bell and T. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[18] C. BenAbdelKader and P. Griffin. A local region-based approach to gender classification from face images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[19] Y. Bengio, Y. LeCun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, space displacement neural networks and hidden markov models. In *Advances in Neural Information Processing Systems*, volume 6, pages 937–944. Morgan Kaufmann, 1994.

[20] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Yee-Whye Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces in the news. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 848–854, 2004.

[21] D. Beymer and T. Poggio. Face recognition from one example view. AI Memo 1536, MIT AI Lab, 1995.

[22] The BioID face database. *http://www.humanscan.com/support/downloads /facedb.php*.

[23] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh ACM, 1992.

[24] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[25] A. Bouzerdoum. The elementary movement detection mechanism in insect vision. In *Philosophical Transactions: Biological Sciences*, volume B-339, pages 375–384, 1993.

[26] A. Bouzerdoum. Classifiation and function approximation using feed-forward shunting inhibitory artificial neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 613–618, 2000.

[27] A. Bouzerdoum and R. B. Pinter. Nonlinear lateral inhibition applied to motion detection in the fly visual system. In R. B. Pinter and B. Nabet, editors, *Nonlinear Vision*, pages 423–450. Boca Raton, 1992.

[28] A. Bouzerdoum and R. B. Pinter. Shunting inhibitory cellular neural networks: Derivation and stability analysis. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 40(3):215–221, March 1993.

[29] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

[30] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:312–355, 1988.

[31] L. M. Brown and Y. L. Tian. Comparative study of coarse head pose estimation. *IEEE Workshop on Motion and Video Computing*, 6, December 2002.

[32] R. Brunelli and T. Poggio. HyperBF networks for gender classification. In *DARPA Image Understanding Workshop*, pages 311–314, 1992.

[33] Roberto Brunelli and Tomaso Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[34] S. Buchala, N. Davey, R. J. Frank, and T. M. Gale. Dimensionality reduction of face images for gender classification. In *Proceedings of the IEEE Conference on Intelligent Systems*, 2004.

[35] J. F. Cardoso. Higher-order contrasts for independent component analysis. *Neural Computation*, 11(1):157–192, 1999.

[36] H. Çevikalp, M. Neamtu, M. Wilkes, and A. Barkana. Discriminative common vectors for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):4–13, 2005.

[37] L. W. Chan and F. Fallside. An adaptive learning algorithm for back-propagation networks. *Computer Speech and Language*, 2:205–218, 1987.

[38] Qian Chen, Haiyuan Wu, T. Fukumoto, and M. Yachida. 3D head pose estimation without feature tracking. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 88–93, April 1998.

[39] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2005.

[40] A. J. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[41] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[42] T. F. Cootes and C. J. Tayler. Locating faces using statistical feature detectors. In *International Conference on Automatic Face and Gestures Recognition*, pages 204–211, Los Alamitos, California, October 1996. IEEE Computer Society Press.

[43] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models–their training and application. *Computer Vision Graphics and Image Understanding*, 61(1):38–59, 1995.

[44] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[45] G. Cottrell and M. Fleming. Face recognition using unsupervised feature extraction. In *Proceedings of the International Conference on Neural Networks*, pages 322–325, 1990.

[46] G. Cottrell and J. Metcalfe. EMPATH: Face, emotion, and gender recognition using holons. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 564–571. Morgan Kaufmann, 1991.

[47] G. Cottrell, P. Munro, and D. Zipser. Learning internal representations from gray-scale images: an example of extensional programming. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, pages 462–473, Seattle, WA, 1987.

[48] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

[49] D. Cristinacce and T. Cootes. A comparison of shape constrained facial feature detectors. In *Proceedings of the 6th International Conference on Automatic Face and Gesture Recognition*, pages 375–380, Seoul, Korea, 2004.

[50] D. Cristinacce, T. Cootes, and I. Scott. A multi-stage approach to facial feature detection. In *Proceedings of the British Machine Vision Conference*, pages 277–286, 2004.

[51] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, pages 327–334, 2004.

[52] Hasan Demirel, Thomas J. Clarke, and Peter Y. K. Cheung. Adaptive automatic facial feature segmentation. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 277–282, 1996.

[53] S. Duffner and C. Garcia. A connexionist approach for robust and precise facial feature detection in complex scenes. In *Fourth International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 316–321, Zagreb, Croatia, September 2005.

[54] S. Duffner and C. Garcia. A hierarchical approach for precise facial feature detection. In *Compression et Représentation des Signaux Audiovisuels (CORESA)*, pages 29–34, Rennes, France, November 2005.

[55] S. Duffner and C. Garcia. A neural scheme for robust detection of transparent logos in TV programs. In *International Conference on Artificial Neural Networks (ICANN)*, volume 2, pages 14–23, Athens, Greece, September 2006.

[56] S. Duffner and C. Garcia. Face recognition using non-linear image reconstruction. In *International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, London, UK, September 2007.

[57] S. Duffner and C. Garcia. An online backpropagation algorithm with validation error-based adaptive learning rate. In *International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 249–258, Porto, Portugal, September 2007.

[58] S. Duffner and C. Garcia. Robust face alignment using convolutional neural networks. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Funchal, Portugal, January 2008. accepted.

[59] S. Duffner and C. Garcia. Robust hierarchical detection of facial features in complex scenes. *Pattern Analysis & Applications*, 2008. accepted.

[60] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition*, pages 300–305, April 1998.

[61] E. Elagin, J. Steffens, and H. Neven. Automatic pose estimation system for human faces based on bunch graph matching technology. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 136–141, April 1998.

[62] M. J. Er, S. Wu, J. Lu, and H. L. Toh. Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks*, 13:697–709, 2002.

[63] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, editors, *Connectionist Models Summer School*, pages 38–51. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[64] B. Fasel. Multiscale facial expression recognition using convolutional neural networks. In *Proceedings of the Third Indian Conference on Computer Vision, Graphics and Image Processing*, Ahmedabad, India, 2002.

[65] B. Fasel. Robust face analysis using convolutional neural networks. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 40–43, Quebec, Canada, 2002.

[66] J.-M. Fellous. Gender discrimination and predication on the basis of facial metric information. *Vision Research*, 37:1961–1973, 1997.

[67] R. Féraud, O. Bernier, J.-E. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, 2002.

[68] R. S. Feris, J. Gemmell, K. Toyama, and V. Krüger. Hierarchical wavelet networks for facial feature localization. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[69] R. A. Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 17:179–188, 1936.

[70] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Second International Conference on Computational Learning Theory*, 1995.

[71] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[72] B. Fröba and C. Küllbeck. Orientation template matching for face localization in complex visual scenes. In *International Conference on Image Processing*, pages 251–254, 2000.

[73] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136, 1975.

[74] K. Fukushima. Neocognitron: A self-organizing neural-network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[75] K. Fukushima. A neural-network model for selective attention in visual pattern recognition. *Biological Cybernetics*, 55:5–15, 1986.

[76] K. Fukushima. Analysis of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2:413–421, 1989.

[77] K. Fukushima and T. Imagawa. Recognition and segmentation of connected characters with selective attention. *Neural Networks*, 6:33–41, 1993.

[78] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognitron. *IEEE Transactions on Neural Networks*, 2(3):335–365, May 1991.

[79] Ken-ichi Funahashi. On the approximate realization of identity mappings by three-layer neural networks. Technical report, Toyohashi University of Technology, Department of Information and Computer Sciences, 1990.

[80] Y. Gao and K. H. Leung. Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):764–779, 2002.

[81] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004.

[82] C. Garcia and S. Duffner. Facial image processing with convolutional neural networks. In *International Workshop on Advances in Pattern Recognition (IWAPR)*, pages 97–108, Plymouth, United Kingdom, July 2007.

[83] C. Garcia, G. Simantiris, and G. Tziritas. A feature-based face detector using wavelet frames. In *Intern. Workshop on Very Low Bitrate Video Coding*, pages 71–76, Athens, 2001.

[84] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, June 2001.

[85] A. Gepperth. Visual object classification by sparse convolutional neural networks. In *Proceedings of the European Symposium on Artificial Neural Networks*, 2006.

[86] B. A. Golomb, D. T. Lawrence, and T. J. Sejnowski. Sexnet: A neural network identifies sex from human faces. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 572–577. Morgan Kaufmann, 1991.

[87] S. Gong, S. McKenna, and J. Collins. An investigation into face pose distributions. In *Proceedings of the 2nd International Workshop on Automatic Face and Gesture Recognition*, pages 265–270, 1996.

[88] Nicolas Gourier, Daniela Hall, and James L. Crowley. Facial feature detection robust to pose, illumination and identity. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[89] A. B. A. Graf and F. A. Wichmann. Gender classification of human faces. *Biologically Motivated Computer Vision*, pages 491–501, 2002.

[90] R. Gross, J. Shi, and J. Cohn. Quo vadis face recognition? In *Third Workshop on Empirical Evaluation Methods in Computer Vision*, December 2001.

[91] S. Grossberg, editor. *Neural Networks and Natural Intelligence*. MIT Press, Cambridge, MA, 1988.

[92] S. Gutta, J. R. J. Huang, P. Jonathon, and H. Wechsler. Mixture of experts for classification of gender, ethnic origin, and pose of human faces. *IEEE Transactions on Neural Networks*, 11(4):948–960, July 2000.

[93] M. Hamouz, J. Kittler, J.-K. Kamarainen, P. Paalanen, and H. Kalviainen. Affine-invariant face detection and localization using GMM-based feature detectors and enhanced appearance model. In *Proceedings of the Sixth International Conference on Automatic Face and Gesture Recognition*, pages 67–72, 2004.

[94] M. Hamouz, J. Kittler, J.-K. Kamarainen, P. Paalanen, H. Kälviäinen, and J. Matas. Feature-based affine-invariant localization of faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1490–1495, 2005.

[95] M.E. Harmon and L.C. Baird III. Multi-player residual advantage learning with general function approximation. Technical report, Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH, 1996.

[96] B. Heisele, P. Ho, J. Wu, and T Poggio. Face recognition: component-based versus global approaches. *Computer Vision and Image Understanding*, 91(1):6–21, 2003.

[97] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. AI Memo 1687, Center for Biological and Computational Learning, MIT, Cambridge, MA, 2000.

[98] E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[99] B. K. P Horn. *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View*. PhD thesis, Massachusetts Institute of Technology, 1970.

[100] B. K. P. Horn and M. J. Brooks. *Shape from Shading*. MIT Press: Cambridge, MA, 1989.

[101] H. J. Hotelling. Analysis of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[102] H.-C. Hsin, C.-C. Li, M. Sun, and R.J. Sclabassi. An adaptive training algorithm for back-propagation neural networks. In *International Conference on Systems, Man and Cybernetics*, volume 2, pages 1049–1052, 1992.

[103] C. Hu, R. Feris, and M. Turk. Active wavelet networks for face alignment. In *British Machine Vision Conference*, UK, 2003.

[104] C. Huang, B. Wu, H. Ai, and S. Lao. Omni-directional face detection based on real adaboost. In *Proceedings of the International Conference on Image Processing*, Singapore, 2004.

[105] J. Huang, X. Shao, and H. Wechsler. Face pose discrimination using support vector machines (SVM). In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, volume 1, pages 154–156, August 1998.

[106] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[107] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

[108] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, New York, 2001.

[109] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[110] K. Ikeuchi and B. K. P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981.

[111] N. Intrator, D. Reisfeld, and Y. Yeshurun. Extraction of facial features for recognition using neural networks. In *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 260–265, Zurich, 1995.

[112] R. A. Jacobs. Increased rates of convergence through learning rate adaption. *Neural Networks*, 1:295–307, 1988.

[113] A. Jain and J. Huang. Integrating independent components and linear discriminant analysis for gender classification. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2004.

[114] N. Japkowitz, S. Hanson, and A. Gluck. Nonlinear autoassociation is not equivalent to PCA. *Neural Computation*, 12(3):531–545, 2000.

[115] S. Jeng, H. Yao, C. Han, M. Chern, and Y. Liu. Facial feature detection using geometrical face model: An efficient approach. *Pattern Recognition*, 31(3):273–282, 1998.

[116] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. In J. Bigun and F. Smeraldi, editors, *Third International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA)*, volume 2091 of *Lecture Notes in Computer Science*, pages 90–95, Halmstad, Sweden, 2001. Springer.

[117] K. Jia, S. Gong, and A. P. Leung. Coupling face registration and super-resolution. In *British Machine Vision Conference*, pages 449–458, Edinburg, UK, September 2006.

[118] M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, Mitsubishi Electric Research Laboratories, 2003.

[119] K. Jonsson, J. Kittler, Y. Li, and J. Matas. Learning support vectors for face verification and recognition. In *Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition*, pages 208–213, 2000.

[120] C. Jutten and J. Herault. Blind separation of sources. *Signal Processing*, 24:1–10, 1991.

[121] J. Keeler, D. Rumelhart, and W. Leow. Integrated segmentation and recognition of hand-printed numerals. *Neural Information processing Systems*, 3:557–563, 1991.

[122] 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*, 12(1):103–108, 1990.

[123] T. Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[124] T. Kohonen. *Self-Organizing and Associative Memory.* Springer-Verlag, Berlin, 1989.

[125] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.

[126] B. Krüger, S. Bruns, and G. Sommer. Efficient head pose estimation with gabor wavelet networks. In *Proceedings of the 11th British Machine Vision Conference*, volume 1, pages 72–81, September 2000.

[127] N. Krüger, M. Pötzsch, and C. von der Malsburg. Estimation of face position and pose with labeled graphs. *Proceedings of the British Machine Vision Conference*, pages 735–743, 1996.

[128] N. Kumar, V. Abhishek, and G. Gautam. A novel approach for person authentication and content-based tracking in videos using kernel methods and active appearance models. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, volume 2, pages 1384–1389, 2003.

[129] M. Lando and S. Edelman. Generalization from a single view in face recognition. In *International Workshop on Automatic Face- and Gesture Recognition*, 1995.

[130] A. Lanitis, C. J. Taylor, and T. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, July 1997.

[131] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13:393–401, 1995.

[132] S. Lawrence, C. Giles, A. Tsoi, and A. Black. Face recognition: A convolutional neural network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

[133] Y. LeCun. Learning processes in an asymmetric threshold network. In E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch, editors, *Disordered systems and biological organization*, pages 233–240. Springer-Verlag, Les Houches, France, 1986.

[134] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufman, Denver, CO, 1990.

[135] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2004.

[136] Y. LeCun, Bottou L., Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[137] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems*. MIT Press, 2005.

[138] Y. LeCun, P. Simard, and B. Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[139] Mun Wai Lee and S. Ranganath. Pose-invariant face recognition using a 3D deformable model. *Pattern Recognition*, 36:1835–1846, 2003.

[140] J. Leonard and M.A. Kramer. Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering*, 14(3):337–341, 1990.

[141] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 637–644, 1995.

[142] B. Q. Li and B. Li. Building pattern classifiers using convolutional neural networks. *International Joint Conference on Neural Networks*, 5:3081–3085, 1999.

[143] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proceedings of the IEEE Conference on European Conference on Computer Vision*, pages 67–81, 2002.

[144] Y. Li, S. Gong, and H. Liddell. Support vector regression and classification based multi-view face detection and recognition. *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2000.

[145] Chun-Hung Lin and Ja-Ling Wu. Automatic facial feature extraction by genetic algorithms. *IEEE Transactions on Image Processing*, 8(6):834–845, 1999.

[146] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural networks. *IEEE Transactions on Neural Networks*, 8:114–132, 1997.

[147] S.-C. B. Lo, S.-L. A. Lou, J.-S. Lin, and M. T. Freedman. Artificial convolutional neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, 14(4):711–718, December 1995.

[148] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece, 1999.

[149] D. G. Luenberger. *Introduction to linear and nonlinear programming*. Addison-Wesley, 1973.

[150] B. MacLennan. Gabor representations of spatiotemporal visual images. Technical Report UT-CS-91-144, Computer Science Department, University of Tennessee, Knoxville, 1991.

[151] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Effective backpropagation with variable stepsize. *Neural Networks*, 10:69–82, 1997.

[152] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

[153] A. M. Martinez. Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):748–763, 2002.

[154] A. M. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, June 1998.

[155] A. M. Martinez and A. C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[156] O. Matan, C. Burges, Y. LeCun, and J. Denker. Multi-digit recognition using a space displacement neural network. In J. Moody, S. Hanson, and R. Lipmann, editors, *Advances in Neural Information Processing Systems 4*, pages 488–495. Morgan Kaufmann, San Mateo, CA, 1992.

[157] M. Matsugu, K. Mori, and Y. Mitari. Convolutional spiking neural network model for robust face detection. In *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP)*, volume 2, pages 660–664, 2002.

[158] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda. Facial expression recognition combined with robust face detection in a convolutional neural network. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2243–2246, July 2003.

[159] S. McKenna and S. Gong. Real time face pose estimation. *International Journal on Real Time Imaging, Special Issue on Real-time Visual Monitoring and Inspection*, 4:333–347, 1998.

[160] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In *Second International Conference on Audio and Video-based Biometric Person Authentication*, pages 72–77, 1999.

[161] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. Müller. Fisher discriminant analysis with kernels. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 41–48, 1999.

[162] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[163] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[164] B. Moghaddam and M.-S. Yang. Learning gender with support faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):707–711, May 2002.

[165] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[166] M. C. Motwani and Qiang Ji. 3D face pose discrimination using wavelets. In *Proceedings of the International Conference on Image Proceedings*, volume 1, pages 1050–1053, 2001.

[167] M. C. Mozer. The perception of multiple objects: A connectionist approach. In *Connectionism in Perspective*. MIT Press-Bradford Books, Cambridge, MA, 1991.

[168] A. Namphol, M. Arozullah, and S Chin. Higher order data compression with neural networks. In *Proceedings of the IJCNN*, pages 155–159, June 1991.

[169] A. V. Nefian. *A Hidden Markov Model-Based Approach for Face Detection and Recognition*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1999.

[170] Claus Neubauer. Evaluation of convolutional neural networks. *IEEE Transactions on Neural Networks*, 9(4):685–696, July 1998.

[171] J. Ng and S. Gong. Multi-view face detection and pose estimation using a composite support vector machine across the view sphere. In *Proceedings. International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 14–21, 1999.

[172] S. Niyogi and W. T. Freeman. Example-based head tracking. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 374–378, October 1996.

[173] S. J. Nowlan and J. C. Platt. A convolutional neural network hand tracker. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 901–908. The MIT Press, 1995.

[174] E. Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 737–745, Espoo, Finland, June 1991.

[175] M. Osadchy, Y. LeCun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215, May 2007.

[176] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[177] A. J. O'Toole, T. Vetter, N. F. Troje, and H. H. Bulthoff. Sex classification is better with three-dimensional structure than with image intensity information. *Perception*, 26:75–84, 1997.

[178] R. Ouellette, M. Browne, and K. Hirasawa. Genetic algorithm optimization of a convolutional neural network for autonomous crack detection. In *Congress on Evolutionary Computation*, volume 1, pages 516–521, June 2004.

[179] V.P. Palgianakos, M.N. Vrahatis, and G.D. Magoulas. Nonmonotone methods for backpropagation training with adaptive learning rate. In *International Joint Conference on Neural Networks*, volume 3, pages 1762–1767, 1999.

[180] P. S. Penev and J. J. Atick. Local feature analysis: a general statistical theory for object representation. *Network: Computation in Neural Systems*, 7(3):477–500, 1996.

[181] A. D. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition*, pages 84–91, 1994.

[182] F. Perronnin, J. L. Dugelay, and K. Rose. A probabilistic model of face mapping with local transformations and its application to person recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1157–1171, 2005.

[183] P. J. Philips, H. Wechsler, J. Huang, and P Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[184] P. J. Phillips. Support vector machines applied to face recognition. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems*, pages 803–809, 1999.

[185] The Psychological Image Collection at Stirling University (PICS). *http://pics.psych.stir.ac.uk*.

[186] D. Plaut, S. Nowlan, and G. E. Hinton. Experiments on learning by back propagation. Technical Report TR CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, 1986.

[187] M. Powell. Radial basis functions for multivariable interpolation: A review. In *IMA Conference on Algorithms for the Approximation of Functions and Data*, 1985.

[188] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[189] J. R. Price and T. F. Gee. Face recognition using direct, weighted linear discriminant analysis and modular subspaces. *Pattern Recognition*, 38(2):209–219, January 2005.

[190] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[191] L. R. Rabiner and Juang B.-H. *Fundamentals of Speech Recognition*. Prentice Hall, Eaglewood Cliffs, NJ, 1993.

[192] R. Rae and H. Ritter. Recognition of human head orientation based on artificial neural networks. *IEEE Transactions on Neural Networks*, 9(2):257–265, March 1998.

[193] Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2007.

[194] M. J. T. Reinders, R. W. C. Koch, and J. J. Gerbrands. Locating facial features in image sequences using neural networks. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 230–235, 1996.

[195] Daniel Reisfeld and Yehezkel Yeshurun. Robust detection of facial features by generalized symmetry. In *Proceedings of the International Conference on Pattern Recognition*, 1992.

[196] E. Rentzeperis, A. Stergiou, A. Pnevmatikakis, and L. Polymenakos. Impact of face registration errors on recognition. In *Artificial Intelligence Applications and Innovations*, Peania, Greece, 2006.

[197] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[198] H. Ritter, T. Martinez, and K. Schulten. *Neuronale Netze*. Addison-Wesley, 1991.

[199] J. Ros, C. Laurent, and G. Lefebvre. A cascade of unsupervised and supervised neural networks for natural image classification. In *International Conference on Image and Video Retrieval*, pages 92–101, Tempe, USA, 2006.

[200] F. Rosenblatt. The perceptron: a probabilistic model for information storage and retrieval in the brain. *Psychological Review*, 65:386–408, 1958.

[201] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, Washington, DC, 1962.

[202] D. Roth, M.-H. Yang, and N. Ahuja. A SNoW-based face detector. In *Advances in Neural Information Processing Systems 12*, pages 855–861. MIT Press, 2000.

[203] S. Roux, F. Mamalet, and C. Garcia. Embedded convolutional face finder. In *International Conference on Multimedia and Expo*, pages 285–288, 2006.

[204] S. Roux, F. Mamalet, C. Garcia, and S. Duffner. An embedded robust facial feature detector. In *International Conference on Machine Learning and Signal Processing (MLSP)*, Thessaloniki, Greece, August 2007.

[205] S. Roweis. EM algorithms for PCA and SPCA. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, volume 10, pages 626–632, 1997.

[206] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[207] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[208] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362. MIT Press, 1986.

[209] Y. Saatci and C. Town. Cascaded classification of gender and facial expression using active appearance models. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 393–398, April 2006.

[210] Eli Saber and A.Murat Tekalp. Frontal-view face detection and facial feature extraction. *Pattern Recognition Letters*, 19(8):669–680, 1998.

[211] Z. Saidane and C. Garcia. Automatic scene text recognition using a convolutional neural network. In *Proceedings of the Second International Workshop on Camera-Based Document Analysis and Recognition (CBDAR)*, September 2007.

[212] Z. Saidane and C. Garcia. Robust binarization for video text recognition. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, September 2007.

[213] Ralf Salomon. Improved convergence rate of back-propagation with dynamic adaptation of the learning rate. In *PPSN*, pages 269–273, 1990.

[214] F. Samaria and A. Harter. Parametrisation of a stochastic model for human face identification. In *2nd IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994.

[215] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the marging: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

[216] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1998.

[217] B. Schiele and A. Waibel. Gaze tracking based on face-color. *International Workshop on Automatic Face and Gesture Recognition*, pages 344–348, 1995.

[218] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 746–751, 2000.

[219] B. Schölkopf, A. J. Smola, and K. R. Müller. Kernel principal component analysis. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[220] N.N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Ninth International Conference on Artificial Neural Networks (ICANN)*, volume 2, pages 569–574, 1999.

[221] I. M. Scott, T. F. Cootes, and C. J. Tayler. Improving appearance model matching using local image structure. In *18th International Conference on Information Processing in Medical Imaging*, pages 258–269, July 2003.

[222] Gregory Shakhnarovich, Paul Viola, and Baback Moghaddam. A unified learning framework for real time face detection and classification. In *International Conference on Automatic Face and Gesture Recognition*, 2002.

[223] T. Shakunaga, K. Ogawa, and S. Oki. Integration of eigentemplate and structure matching for automatic facial feature detection. In *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 94–99, April 1998.

[224] S. Shan, Y. Chang, W. Gao, B. Cao, and P. Yang. Curse of mis-alignment in face recognition: problem and a novel mis-alignment learning solution. In *Automatic Face and Gesture Recognition*, pages 314–320, 2004.

[225] A. Shashua and T. Riklin-Raviv. The quotient image: Class-based rerendering and recognition with varying illuminations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):129–139, February 2001.

[226] F. M. Silva and L. B. Almeida. Acceleration techniques for the backpropagation algorithm. In *Proceedings of the EURASIP Workshop 1990 on Neural Networks*, pages 110–119, London, UK, 1990. Springer-Verlag.

[227] T. Sim, S. Baker, and M. Bsat. The CMU Pose, Illumination and Expression (PIE) database. Technical Report CMU-RI-TR-01-02, The Robotics Institute, CMU, January 2001.

[228] T. Sim and Kanade. T. Combining models and exemplars for face recognition: An illuminating example. In *Workshop on Models versus Exemplars in Computer Vision*, 2001.

[229] S. Sirohey and A. Rosenfeld. Eye detection in a face image using linear and nonlinear filters. *Pattern Recognition*, 34(7):1367–1391, 2001.

[230] Z. Sun, G. Bebis, X. Yuan, and S. J. Louis. Genetic feature subset selection for gender classification: A comparison study. In *IEEE Workshop on Applications of Computer Vision*, pages 165–170, August 2002.

[231] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[232] R. S. Sutton. Adapting bias by gradient descent: an incremental version of belta-bar-delta. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 171–176, Cambridge MA, 1992.

[233] D. L. Swets and J. J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[234] B. Takács. Comparing face images using the modified hausdorff distance. *Pattern Recognition*, 31:1873–1881, 1998.

[235] C. E. Thomaz, R. Q. Feitosa, and A. Veiga. Design of radial basis function network as classifier in face recognition using eigenfaces. In *5th Brazilian Symposium on Neural Networks (SBRN)*, pages 118–123, 1998.

[236] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. Technical Report NCRG-97-010, Aston University, 1997.

[237] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[238] F. H. C. Tivive and A. Bouzerdoum. A new class of convolutional neural network (SICoNNets) and their application to face detection. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2157–2162, 2003.

[239] F. H. C. Tivive and A. Bouzerdoum. Efficient training algorithms for a class of shunting inhibitory convolutional neural networks. *IEEE Transactions on Neural Networks*, 16(3):541–556, May 2005.

[240] F. H. C. Tivive and A. Bouzerdoum. A fast neural-based eye detection system. In *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 641–644, December 2005.

[241] F. H. C. Tivive and A. Bouzerdoum. A shunting inhibitory convolutional neural network for gender classification. In *Proceedings of the International Conference on Pattern Recognition*, 2006.

[242] L. Torres, L. Lorente, and J. Vila. Automatic face recognition of video sequences using self-eigenfaces. In *International Symposium on Image/Video Communication over Fixed and Mobile Networks*, 2000.

[243] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[244] S. Usui, S. Nakauchi, and M. Nakano. Internal color representation acquired by a five-layer neural network. In O. Simula, T. Kohonen, K. Makiiara, and J. Kangas, editors, *Proceedings of the International Conference on Artificial Neural Networks*, pages 867–872, 1991.

[245] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Russia (English Translation: Springer Verlag, New York, 1982), 1979.

[246] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[247] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 511–514, 2002.

[248] T. Vetter and T Poggio. Linear object classes and image synthesis from a single example image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):733–741, 1997.

[249] J. M. Vincent, J. B. Waite, and D. J. Myers. Precise location of facial features by a hierarchical assembly of neural nets. In *Second International Conference on Artificial Neural Networks*, pages 69–73, 1991.

[250] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 511–518, 2001.

[251] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[252] M. Visani, C. Garcia, and J. M. Jolion. Two-dimensional-oriented linear discriminant analysis for face recognition. In *Proceedings of the International Conference on Computer Vision and Graphics*, pages 1008–1017, 2004.

[253] M. Visani, C. Garcia, and J. M. Jolion. Normalized radial basis function networks and bilinear discriminant analysis for face recognition. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, Como, Italy, September 2005.

[254] M. Visani, C. Garcia, and C. Laurent. Comparing robustness of two-dimensional PCA and eigenfaces for face recognition. In A. Campilho and M. Kamel, editors, *Proceedings of the International Conference on Image Analysis and Recognition*, pages 717–724, 2004.

[255] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink, and D.L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59:257–263, 1988.

[256] Yucheng Wei, L. Fradet, and Tieniu Tan. Head pose estimation using gabor eigenspace modeling. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 281–284, 2002.

[257] P Werbos. Backpropagation: Past and future. In *Proceedings of the International Conference on Neural Networks*, pages 343–353, 1988.

[258] D. J. Willshaw and C. von der Malsburg. How patterned neural connections can be set up by self-organization. In *Royal Society of London*, volume 194 of *Series B-194*, pages 431–445, 1976.

[259] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg. Face recognition and gender determination. In *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 92–97, 1995.

[260] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[261] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:913–931, 2003.

[262] O. Yamaguchi, K. Fukui, and K. I. Maeda. Face recognition using temporal image sequence. In *Proceedings of the IEEE Conference on Face and Gesture Recognition*, pages 318–323, 1998.

[263] J. Yang, D. Zhang, and A. F. Frangi. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004.

[264] M. H. Yang. Kernel Eigenfaces vs. Kernel Fisherfaces: Face recognition using kernel methods. In *Proceedings of the International Conference on Face and Gesture Recognition*, pages 215–220, 2002.

[265] M.-H. Yang, D. Kriegman, and N. Ahuja. Face detection using multimodal density models. *Computer Vision and Image Understanding*, 84:264–284, 2001.

[266] M. H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[267] B. Yegnanarayana, Anil Kumar sao, B. V. K. V. Kumar, and M. Savvides. Determination of pose angle of face using dynamic space warping. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, volume 1, pages 661–664, April 2004.

[268] K. C. Yow and R. Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713–735, 1997.

[269] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[270] L. Zhao, G. Pingali, and I. Carlbom. Real-time head orientation estimation using neural networks. In *Proceedings of the International Conference on Image Processing*, September 2002.

[271] W. Zhao. *Robust Image Based 3D Face Recognition*. PhD thesis, University of Maryland, 1999.

[272] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–458, 2003.

[273] S. Zhou, R. Chellappa, and D. Jacobs. Characterization of human faces under illumination variations using rank, integrability, and symmetry constraints. *European Conference on Computer Vision*, pages 588–601, 2004.

[274] M. Zobel, A. Gebhard, D. Paulus, J. Denzler, and H. Niemann. Robust facial feature localization by coupled features. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 2–7, 2000.