Neural learning methods for visual object detection

(Neuronale Lernverfahren zur visuellen Objekterkennung)

Dissertation

zur Erlangung des Grades
"Doktor der Naturwissenschaften"
an der Fakultät für Physik und Astronomie
der Ruhr-Universität Bochum

vorgelegt von
Alexander Rainer Tassilo Gepperth
am 19.April 2006

Datum der Disputation: 5.7.2006
Gutachter: Prof. Dr. Gregor Schöner und Prof. Dr. Christoph von der Malsburg

Acknowledgments

This thesis summarizes the greater part of the work I did during my three years at the Institute for Neural Dynamics in Bochum. Scientific work is a collaborative effort; I am therefore indebted to many of my colleagues for their friendship, inspirations, constructive criticism and help in practical matters. I greatly enjoyed my time in Bochum because of the open and friendly atmosphere at the institute, which made me feel at home almost instantly. Credits for this must go to Prof. Werner von Seelen, the former head of the institute, and to the current head Prof. Gregor Schöner.

Especially, I wish to thank Hannes Edelbrunner for his continuing support, beer and inspiring midnight discussions, Christian Igel for valuable neural network tips, and Michael Neef for making backups of all the files I could possibly delete. I am grateful to Stefan Roth for good and reliable partnership in our common projects, and our secretaries Ute Kopp and Angelika Wille for good humor and help with university forms.

Furthermore, I wish to thank my proofreaders Britta Mersch, Thorsten Suttorp, Jan Salmen, Hannes Edelbrunner and Markus Mildenstein without whose help this thesis would have been handed in half a year later (or, alternatively, full of "German English"). I wish to thank Britta Mersch for encouraging me and believing in me during the last few uncomfortable months, and also for the fact that she exists. Thanks go to my relatives Hanne and Hans Hambücher for their continuing moral and material support.

This thesis is dedicated to my parents.

Contents

[1] Some of the content of this chapter has been published in: A. Gepperth, J. Edelbrunner, and T. Bücher. Real-time detection and classification of cars in video sequences. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 625–631, 2005.

[2]Some of the content of this chapter has been published in A. Gepperth. Visual object classification by sparse convolutional networks. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2006*. d-side publications, 2006. accepted.

[3]Some of the content of this chapter has been published in: A. Gepperth and S. Roth. Applications of multi-objective structure optimization. *Neurocomputing*, (69):701–713, 2006. Design and implementation of the evolutionary multi-objective optimization algorithm and the implementation of multi-objective performance measures was done by S.Roth.

Comments on notation and abbreviations

For clarity's sake, each chapter defines the abbreviations it uses even if they were defined in previous chapters already. Some common abbreviations are

- NN – neural network
- MLP – multilayer perceptron
- ROI – region of interest
- CNN – convolutional neural network
- SCNN – sparse convolutional neural network
- DFT – discrete Fourier transform
- MOO – multi-objective optimization
- MSE – mean squared error
- CE – classification error
- CMA – covariance matrix adaptation
- SVM – support vector machine
- RF – receptive field

Mathematical notation is defined wherever it is required: some common conventions are:

- Vectors are denoted by bold characters $\boldsymbol{z}$ and their components by superscripts: z^1.
- Transposition of vectors is indicated by $\boldsymbol{z}^T$.
- The norm of vectors $\boldsymbol{z}$ is written $||\boldsymbol{z}||$.
- The imaginary number $\sqrt{-1}$ is denoted i.

- The Kronecker symbol is denoted by δ_{ij} and is nonzero only if $i = j$
- The two-dimensional delta distribution is denoted $\delta(x, y)$.
- The Fourier transform of a function is written in calligraphic letters: $\mathcal{F}(\omega)$.

Introduction

Visual object detection and classification are tasks that are getting increasingly important in everyday technological applications. Some examples are face detection systems for video surveillance, video conferencing, advanced image compression, traffic monitoring and advanced driver assistance systems.

Despite the apparent ease with which humans solve these tasks, they are still challenging research topics, and human performance has not been reached by a long way. Despite many interesting applications that could be realized using object detection systems that equal human performance, the problem is of considerable academic interest in its own right. Quite often, research is inspired by visual information processing strategies in the human brain, requiring close collaboration of scientists from psychology, neurobiology and computer science. This thesis presents ways to incorporate the principle of *adaptivity*, which is ubiquitous in the human brain, into different stages of the object detection process.

Usually, object detection systems consist of three distinct processing steps: initial detection, feature extraction and classification (see fig. 1). The initial detection stage identifies regions which contain object "candidates" (so-called *regions of interest*—ROIs) using a variety of methods that usually depend on the object class at hand. The obtained ROIs are then subjected to a *feature extraction* step in which certain visual properties or *features* within the ROI are computed. Again, the choice of features is usually task-dependent. As a last step, a trainable object classifier is fed with the computed features and a decision is made whether the content of an ROI is indeed an object belonging to the desired class. Since initial detection has to scan a whole image for ROIs, it is far more time critical than feature extraction or classification, because those steps process only ROIs which are considered likely to contain objects by the initial detection. Moreover, initial detection methods must cope with a very large and complex problem class: distinguishing "objects" from "everything else". Especially the "everything else"-parts of an image can exhibit strong variability, and this a reason why initial detection is seldom perfect. Indeed, perfect initial detection is not a requirement; it is only required that no objects are missed, since missed objects cannot be processed by the subsequent stages at all.

Summarizing the last paragraph, it can be stated that initial detection methods must solve hard classification problems in (usually) limited processing time.

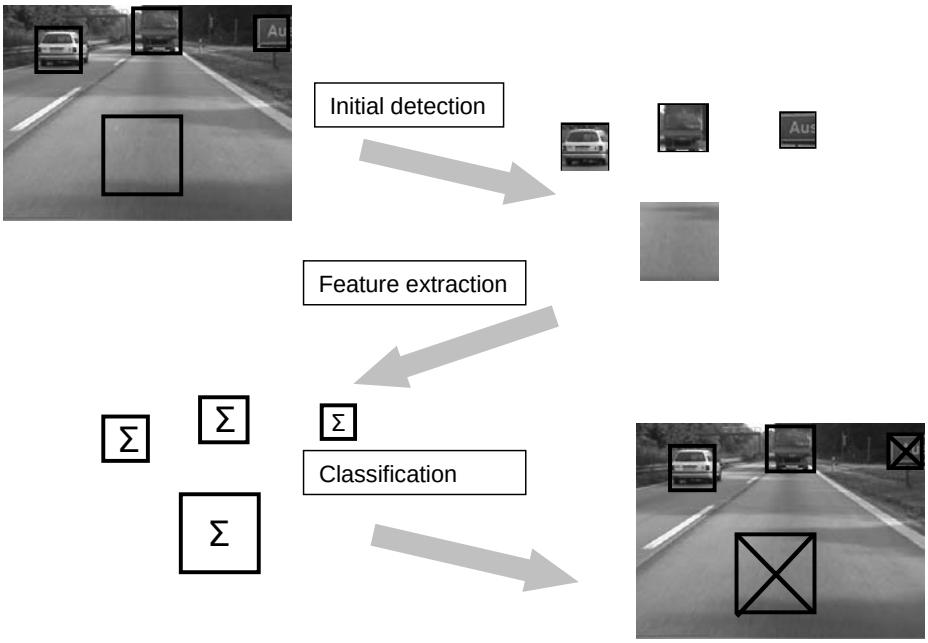


Figure 1: Standard architecture for visual object detection. The Σ characters are meant to indicate that a mathematical transformation has been applied to obtain visual object features. All details of this architecture are subject to considerable variability depending on the area of application and the constraints that must be met.

For this reason, initial detection is often not performed by fully trainable systems like support vector machines or neural networks, but by manually designed methods tailored to suit only a certain object class, both for reasons of computational efficiency and classification accuracy.

Similar comments apply when discussing the feature extraction stage: many models apply a fixed feature extraction designed to represent a certain object class well. For example, when detecting cars, it is well known that strong horizontal and vertical gradients are typical of this object class, so it makes sense to design a feature extraction scheme that in some way reflects the strength of these gradients in its output. It is known that suitable feature extraction can improve classification accuracy significantly; unfortunately, the reverse is also true: unsuitable feature extraction can make classification hard or impossible.

The decision whether an ROI actually contains an object or not is usually taken by a trainable classifier. There are several possible methods that can be used here, among them Linear Discriminant Analysis [36], Bayesian classifiers [33], support vector machines [6, 13], neural networks in different variants [103] or decision trees [9]. All these classifiers can be trained by supplying examples which should define the unknown true classification function. In this thesis,

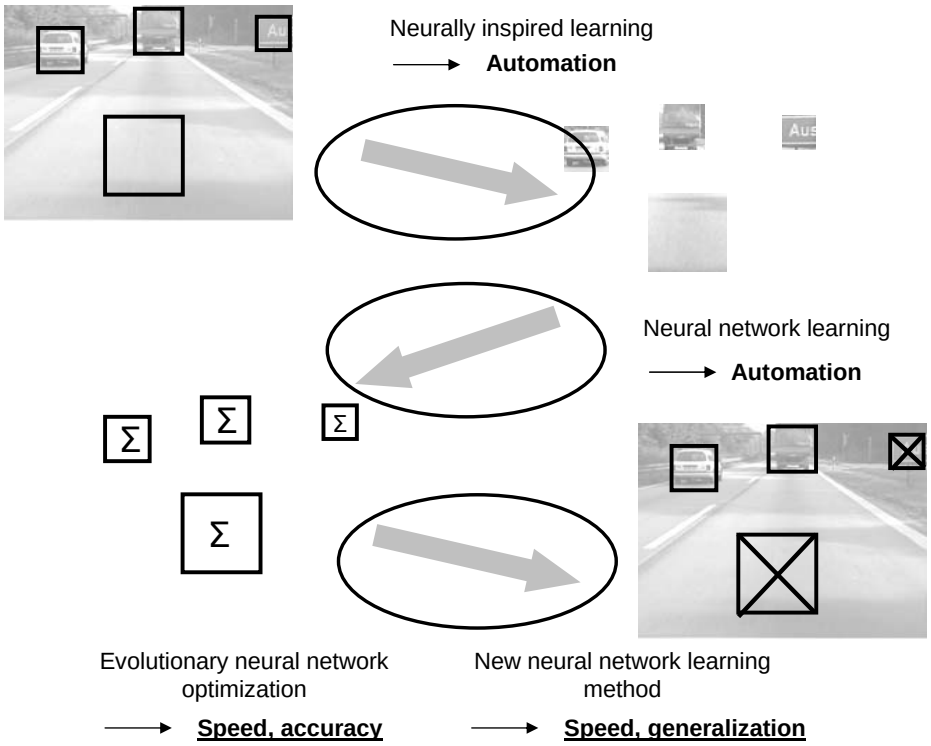


Figure 2: Overview over the improvements to the object detection process proposed in this thesis. Please compare to fig. 1 to get a clear picture.

object classifiers are always implemented by neural networks, although some of the employed network models differ significantly.

Goals and outline

The work described in this thesis aims to improve several, sometimes complementary problems found in object detection and classification. Fig. 2 shows the aspects of the object detection process to which this thesis makes contributions.

The investigation described in chapter 4 proposes an object detection architecture similar to that sketched in fig. 1. It shows the possibilities but also the problems (feature design, network design, initial detection design) that are associated with this kind of architecture. Each of the following chapters makes contributions towards improving a certain component described in chapter 4.

An important goal of the thesis was to replace manually designed algorithms by neural learning wherever possible; what is more, in several cases it is shown that neural learning is not only successfully applicable but also perfectly compatible with real-time and accuracy constraints (chapters 4 and 6) which were

previously thought to require specialized solutions. This mainly applies to the initial detection and feature extraction stages.

In chapter 5, an trainable method for initial detection is proposed. Although it does not use neural networks as such, the model employs neural processing principles like a layered structure, center-surround interactions and biased competition by top-down modulation. It is furthermore inspired by the principle of feature-based attention and can learn simple feature properties of target objects.

In chapter 6 it is demonstrated that one can use neural learning to reduce the initial detection problem to a classification problem: a fast neural classifier is applied at all image locations and several spatial scales. Thus, one could eliminate initial detection altogether from the object detection process. Chapter 6 furthermore shows how to define a meaningful feature extraction in a systematic way using the neural network learning process. The complexity of the object detection problem can be significantly reduced by the neural learning methods presented in chapter 6.

For improving the classification step itself, two different techniques for neural network structure optimization are compared in chapter 7. One of them is a new evolutionary method which – although complicated – is demonstrated to produce neural networks whose structure is well suited to solve a given classification tasks. The other method is a simple pruning heuristic known as *magnitude-based pruning*, which is used as a baseline for comparison. Both optimization methods are evaluated according to multi-objective performance indicators, the objectives being speed and classification accuracy. The practical benefit is that a set of solutions is produced, each of which is optimal with respect to a certain trade-off between the two objectives. One can thus choose an optimal neural network structure that fits the given constraints, but (and this is the point) this can happen *after* the optimization is finished.

Due to the focus of the research group I was working with, most experimental tests are performed on classification problems arising from advanced driver assistance applications, like car detection and traffic sign detection (on one occasion, a face detection problem is investigated).

This thesis is intended for readers who are familiar with neural networks and statistical learning theory. A comprehensive knowledge of digital image processing, however, is not required. Those concepts and techniques which are used in later parts are briefly reviewed in the first part of the thesis. A complete introduction to the subject is neither intended nor, indeed, possible within the scope of a thesis; the interested reader is referred to textbooks like [69] or [106]. I feel it is a justified approach to treat digital image processing in this rather brief way since it is just a tool, not a central issue of this thesis.

In order to enable the reader to assess the contributions of this thesis properly, an overview over selected current object detection architectures which were sources of inspiration is given. Furthermore, since a part of the thesis which I consider very important makes use of a technique called *principal components analysis*, a review with its focus on possible extensions and generalizations has also been included.

Part I

Foundations

Chapter 1

Basic techniques of digital image processing

In this chapter, basic concepts from digital image processing are reviewed and explained. Emphasis is given to those topics which are required for an understanding of the later chapters in part II of this thesis.

1.1 The sampling theorem

The sampling theorem [79, 90, 113, 130] is a mathematical theorem about the representation of continuous functions by discrete samples. It is of tremendous importance in digital image processing because a digital image approximates the "real" image it is derived from by discrete sample points. Therefore, care must be taken when performing operations on digital images not to violate the conditions imposed by the sampling theorem and thereby to corrupt the information contained in the image.

The following section derives the theorem's fundamental statements and then goes on to discuss the implications for digital image processing.

Consider an (infinitely extended) continuous, two-dimensional image described by $F(x, y)$. Sampling means reducing $F(x, y)$ to a discrete (although still infinitely extended) set of sample points. It is demanded that the original image should be recoverable from those points without errors.

Sampling is performed by means of the *delta comb*

$$S(x, y) = \sum_{j_1=-\infty}^{\infty} \sum_{j_2=-\infty}^{\infty} \delta(x - j_1 \Delta x, y - j_2 \Delta y) \ . \tag{1.1}$$

The sampled image is then given as

$$F_S(x,y) = F(x,y)S(x,y) = \sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty} F(x-j_1\Delta x, y-j_2\Delta y) \\ S(x-j_1\Delta x, y-j_2\Delta y) \ . \tag{1.2}$$

When analyzing $F_S(x,y)$, it is useful to change to the spatial frequency domain by applying the two-dimensional Fourier transform. The Fourier transforms of the original image $F(x,y)$ and the sampled image $F_S(x,y)$ are given by

$$\begin{aligned}\mathcal{F}(\omega_x,\omega_y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(x,y)\exp\{-\mathrm{i}(\omega_x x+\omega_y y)\}\mathrm{d}x\mathrm{d}y \\ \mathcal{F}_S(\omega_x,\omega_y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_S(x,y)\exp\{-\mathrm{i}(\omega_x x+\omega_y y)\}\mathrm{d}x\mathrm{d}y \ .\end{aligned} \tag{1.3}$$

$F_S(x,y)$ is obtained by multiplication with the delta comb $S(x,y)$. The Fourier transform of $S(x,y)$ is given by

$$\mathcal{S}(\omega_x,\omega_y) = \frac{4\pi^2}{\Delta x\Delta y}\sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty}\delta(\omega_x-j_1\omega_x^s,\omega_y-j_2\omega_y^s) \ , \tag{1.4}$$

where $\omega_x^s = \frac{2\pi}{\Delta x}$, $\omega_y^s = \frac{2\pi}{\Delta y}$ represent the sampling frequencies in the spatial frequency domain. Using the Fourier transform convolution theorem (see, e.g., [7]) which states that a multiplication in the space domain corresponds to a convolution (see section 1.2 for a formal definition) in the frequency domain, we obtain

$$\mathcal{S}(\omega_x,\omega_x) = \frac{1}{\Delta x\Delta y}\int\int_{-\infty}^{\infty}\mathcal{F}(\omega_x-\alpha,\omega_y-\beta) \\ \sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty}\delta(\alpha-j_1\omega_x^s,\beta-j_2\omega_y^s)\mathrm{d}\alpha\mathrm{d}\beta \ .$$

This expression can be rewritten by applying elementary δ gymnastics (see, e.g., [87] for a summary of the properties of the delta distribution) to give the final result

$$\mathcal{F}_S(\omega_x,\omega_y) = \frac{1}{\Delta x\Delta y}\sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty}\mathcal{F}(\omega_x-j_1\omega_x^s,\omega_y-j_2\omega_y^s) \ . \tag{1.5}$$

This is an extraordinary result: it states that the spectrum of the sampled image is repeated infinitely often at intervals of (ω_x^s,ω_y^s). This is visualized in fig. 1.1. From the figure, it is also obvious that this is only possible if $\mathcal{F}(\omega_x,\omega_y)$ is *band-limited* in the spatial frequency domain. This means that there exist frequencies $\omega_x^{\max},\omega_y^{\max}$ such that $\mathcal{F}(\omega_x,\omega_y) = 0 \ \ \forall(\omega_x,\omega_y) > (\omega_x^{\max},\omega_y^{\max})$. Otherwise, the periodical repetitions of $\mathcal{F}(\omega_x,\omega_y)$ would overlap and it can be shown that

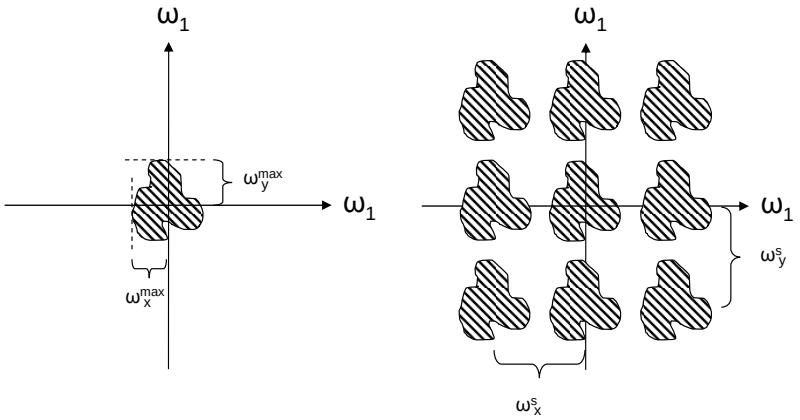


Figure 1.1: Effect of spatial sampling on representation in the frequency domain. The original band-limited spectrum is periodically repeated at intervals given by the sampling frequencies ω_x^s, ω_y^s. If the sampling frequencies are chosen too small, the replicated spectra will overlap, making reconstruction from samples impossible.

exact reconstruction of the original image $F(x, y)$ becomes impossible. Simply speaking, this result expresses that the choice of sampling frequencies is not arbitrary: The correct frequencies depend on the image content in $F(x, y)$ by

$$\begin{aligned} \omega_x^{\max} &\leq \frac{\omega_x^s}{2} \\ \omega_y^{\max} &\leq \frac{\omega_y^s}{2} \quad . \end{aligned} \tag{1.6}$$

This can be expressed as

$$\begin{aligned} \Delta x &\leq \frac{\pi}{\omega_x^{\max}} \\ \Delta y &\leq \frac{\pi}{\omega_y^{\max}} \quad . \end{aligned}$$

In digital image processing, images are always sampled at intervals of one pixel. Although they are of finite size, the validity of the statements given above is not affected, because one can always consider infinite images which have nonzero values in a finite region only. Assuming that the digital image has been correctly sampled, two interesting statements can be derived from the given inequalities:

- The highest possible frequency in the digital image is π/pixel or half a wavelength per pixel.
- When re-sampling an image to a smaller size, frequencies which cannot be represented any longer (since the sampling frequencies ω_x^s and ω_y^s have decreased) must be removed from the image

1.2 Image processing using convolution filters

In digital image processing, *convolutions* of digital images with discretized functions called *convolution filters* play a very important role, as shall be explained in later sections of this chapter. For now, it is sufficient to state that the convolution of a digital image with a convolution filter is an operation that changes the content of the image in a defined way. In order to derive a quantitative description of this process, we will proceed in the same way as in section 1.1 and start from a continuous representation of the image $F(x, y)$ and the convolution filter $C(x, y)$.

Mathematically, a convolution is an operation on function spaces H (for example, $\mathcal{L}^2$): $H \times H \rightarrow H$ which is defined by $(C*F)(\boldsymbol{z}) = \int_D C(\boldsymbol{\alpha})F(\boldsymbol{z}-\boldsymbol{\alpha})\mathrm{d}\boldsymbol{\alpha}$ whenever this integral exists. The support of both functions is denoted by D, $\boldsymbol{z}, \boldsymbol{\alpha} \in D$ and $F(\boldsymbol{z})$, $C(\boldsymbol{z}) \in H$. In the case of two-dimensional images considered here, D corresponds to $\mathbb{R}^2$.

1.2.1 General properties

The following general properties of convolutions are of relevance for digital image processing:

- Linearity

$$\{kC_1(\boldsymbol{z}) + jC_2(\boldsymbol{z})\} * F(\boldsymbol{z}) = kC_1(\boldsymbol{z}) * F(\boldsymbol{z}) + jC_2(\boldsymbol{z}) * F(\boldsymbol{z}) \tag{1.7}$$

- Commutativity

$$C_1(\boldsymbol{z}) * C_2(\boldsymbol{z}) = C_2(\boldsymbol{z}) * C_1(\boldsymbol{z}) \tag{1.8}$$

- Associativity

$$C_1(\boldsymbol{z}) * \{C_2(\boldsymbol{z}) * F(\boldsymbol{z})\} = \{C_1(\boldsymbol{z}) * C_2(\boldsymbol{z})\} * F(\boldsymbol{z}) \tag{1.9}$$

Another very useful property arises from the Fourier transform convolution theorem (see last section) which states that a convolution of two functions amounts to a multiplication of their Fourier transforms. Thus, the effect of a convolution can be very easily analyzed in frequency space. It shall now be demonstrated that the properties that were just stated continue to apply when image and convolution filter are sampled.

1.2.2 Convolutions of sampled functions

Let $F_S(x, y)$ stand for the sampled image and $C_S(x, y)$ for the sampled convolution filter in accordance with eqn. (1.2). It is assumed that sampling happens at a common sampling frequency ω^s in both dimensions and that the conditions (1.6) imposed by the sampling theorem are respected. Then, the Fourier transforms of $F_S(x, y)$ and $C_S(x, y)$ can be computed; by inserting the definitions of

the sampled functions and applying simple properties of the delta distribution, the following expressions are obtained:

$$\begin{aligned}\mathcal{C}_S(\omega_x,\omega_y) &= \sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty} C(j_1\Delta x, j_2\Delta y)\exp\{-\mathrm{i}(\omega_x j_1\Delta x+\omega_y j_2\Delta y)\}\\ \mathcal{F}_S(\omega_x,\omega_y) &= \sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty} F(j_1\Delta x, j_2\Delta y)\exp\{-\mathrm{i}(\omega_x j_1\Delta x+\omega_y j_2\Delta y)\}\ .\end{aligned} \tag{1.10}$$

Now, eqns. (1.10) are just the expressions for the discrete Fourier transform! And, since the Fourier transform convolution theorem continues to hold, it can be stated that convolutions of sampled images with sampled filters amount to a component-wise multiplication of their discrete Fourier transform (DFT) coefficients.

1.2.3 Image processing with discrete convolution filters

At this point it is helpful to recall the information encoded in the Fourier representation of an image or a filter. It is this: High frequencies correspond to fine image details and low spatial frequencies to coarse ones. Thus, the representation of a filter in frequency space directly expresses its effect on the image content.

This insight is of tremendous importance, because it justifies the design of discrete convolution filters directly in frequency space; the position space representation can be computed by inverting the DFT.

In the same way as in (1.10), the convolution formula given above is discretized by the sampling process:

$$\begin{aligned}(C_S * F_S)(\boldsymbol{z}) &= \int_D C_S(\boldsymbol{\alpha})F_S(\boldsymbol{z}-\boldsymbol{\alpha})\mathrm{d}\boldsymbol{\alpha} =\\ &= \sum_{j_1=-\infty}^{\infty}\sum_{j_2=-\infty}^{\infty} C_S(j_1\Delta x, j_2\Delta y)F_S(x-j_1\Delta x, y-j_2\Delta y)\ ,\end{aligned} \tag{1.11}$$

where the second row is obtained by performing the integrals and specializing to two dimensions. The only obstacle to actually using discrete convolutions for image processing is the infinite size of the sampled images and convolution filters, expressed by the infinite sums in eqn. (1.11). Since digital images are usually not derived from a continuous, infinite representation but are already available in a sampled, finite form, they do not cause problems. However, discrete convolution filters often *are* obtained in this way; therefore, they must be made finite by introducing a cut-off length δ: every filter point outside an interval defined by δ is set to zero. This operation is not without its problems because it will introduce additional frequencies into the spectrum of a filter. Treating the

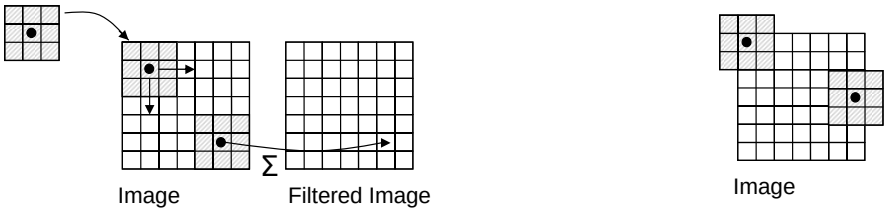


Figure 1.2: Left: Visualization of a discrete convolution. The convolution filter is placed with its center (indicated by the black dot) over a certain image pixel such that each filter pixel is "on top" of one image pixel. Each pair of filter and image pixels is multiplied and the results are summed up, giving one number per image pixel. This number is stored in the transformed image which has the same dimensions as the original image. Right: Boundary effects occurring when parts of the filter mask exceed the image dimensions. Different ways to deal with this situation are described in the text.

problem in one dimension for simplicity, a cut-off corresponds to the component-wise multiplication of the filter with the *window function* $W^{\delta}_{\mathrm{rect}}$ given in (1.12). Due to the Fourier transform convolution theorem, the DFT of the new filter will be a discrete convolution of the original filter transform with the DFT of the window function

$$W^{\delta}_{\mathrm{rect}}(x) = \begin{cases} 1 & |x| < \delta/2 \\ 0 & \text{otherwise} \end{cases}$$
$$\mathcal{W}^{\delta}_{\mathrm{rect}}(k) = \delta \frac{\sin(k\delta/2)}{k\delta/2} \quad , \tag{1.12}$$

and it is obvious that a convolution with $\mathcal{W}^{\delta}_{\mathrm{rect}}(k)$ will, in general, change the DFT significantly.[1] A rule-of thumb for circumventing this problem is to apply the cut-off in regions where the convolution filter is already close to zero. Nevertheless, the effect on the DFT should always be checked.

A finite, discretized convolution filter is also called a *filter mask*, and its DFT is termed a *transfer function*. The process of convolving a digital image with a sampled, finite filter mask is depicted in fig. 1.2 (left).

1.2.4 Practical considerations

In order to apply the techniques described in the last sections in practice, aspects of computational efficiency, the finiteness of image and convolution filters and the general problem of filter design must taken into account.

[1] This is consistent, because as $\delta \to \infty$, $\mathcal{W}^{\delta}_{\mathrm{rect}}(k)$ tends to a delta distribution and the convolution has no effect.

Guidelines for filter design

Filter masks may be designed in a variety of ways [69] to perform desired operations on an image. Two obvious possibilities are the design of filters in position and in frequency space. If filter design is performed in frequency space, a continuous transfer function must be constructed and transformed to position space, where it is sampled and made finite (if necessary). Conversely, a continuous function in position space can serve as a model for the filter mask, which is obtained by sampling and a suitable cut-off.

Boundary conditions

Since images and filter masks are finite, it is possible to apply a filter mask at positions where parts of it exceed the image dimensions. Examples are shown in fig. 1.2 (right). Several possible strategies can be used when this happens:

- *Exclude borders*: The filter mask is not applied and a value of zero is returned.
- *Zero-padding*: Only the part of the filter mask that is within the image is applied.
- *Periodic boundary conditions*: The problem is circumvented by treating the image as infinitely large but periodic in all directions.
- *Mirror boundary conditions*: The image is replicated periodically; in contrast to periodic boundary conditions, the image is mirrored at each boundary such that pixel values never change when crossing a boundary.

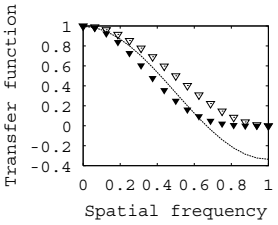
When calculating the transfer functions of filter masks, the frequency is usually normalized by the highest possible frequency (half a wavelength per pixel) that can appear in a digital image. This frequency is commonly called the *Nyquist frequency*.

Separable and non-separable convolution filters

In practice, huge performance gains can be achieved by using *separable filters*. These are filter masks f that can be expressed by discrete convolutions of one-dimensional masks, e.g.: $f = f_x * f_y$. If the filter mask is derived from a continuous function $f(x, y)$, the condition $f(x, y) = f_x(x)f_y(y)$ is sufficient for the separability of the filter mask f. Using (1.8) and (1.9), one can show that

$$f * F = (f_x * f_y) * F = f_x * (f_y * F) = f_y * (f_x * F) \ ,$$

which means that the convolution with filter mask f can be expressed by successive convolutions with the one-dimensional masks f_x and f_y. Please observe that the complexity in the latter case grows linearly with the size of f, whereas it grows polynomially (in two dimensions: quadratically) in the former case.



Order	filter mask	normalization const.
2	1 2 1	4
4	1 4 6 4 1	16
6	1 6 15 20 15 6 1	64
8	1 8 28 56 70 56 28 1	256
n	1,..(n times)	n

Figure 1.3: One-dimensional filter masks and transfer functions of lowpass filters. Left: transfer functions of a box filter of order 3 (solid line), a binomial filter of order 2 (bright triangles), and a binomial filter of order 4 (dark triangles). It is notable that the binomial filters achieve much better smoothing because high frequencies are uniformly attenuated, whereas the box filter allows the highest frequencies to pass with inverted sign. The transfer functions of higher-order binomial filters fall off more quickly, which enables them to filter out lower frequencies. All frequencies are normalized by the Nyquist frequency.

1.3 High- and lowpass filters

A very commonly used class of filters is the class of *smoothing filters*. They remove high frequencies from an image and are therefore *lowpass filters*. It is always possible to construct a highpass filter from a lowpass filter by subtracting a lowpass filter from the identity filter whose entries are equal to zero except for the central pixel which is equal to one.

The class of *binomial filters* approximates a Gaussian function but avoids sampling and cut-off problems (see fig. 1.3). Transfer functions are nonnegative and monotonically decreasing, ensuring that high frequencies are eliminated efficiently.

1.4 Bandpass filters

By considering the transfer functions shown in fig. 1.3, one can perceive that the difference of two distinct lowpass filters (e.g., two binomial filters) will be the transfer function of a *bandpass filter*. Bandpass filters allow only certain frequencies in the image to "survive" the convolution, thus effectively eliminating all image structures outside a certain size range. This behavior can again be read off directly from the transfer function. Fig. 1.4 shows the transfer function of a typical bandpass filter, and fig. 1.7 gives an example for bandpass filtering. The most commonly used bandpass filters are so-called *difference-of-Gaussians* (doG) filters. As their name indicates, they are formed by subtracting the filter masks derived (by sampling and appropriate cut-off) from Gaussian curves with mean zero and different standard deviations. By varying the standard deviations of the two filter masks, different frequency selectivities can be achieved. This is especially easy to calculate since the Fourier transform of a Gaussian is also a Gaussian.

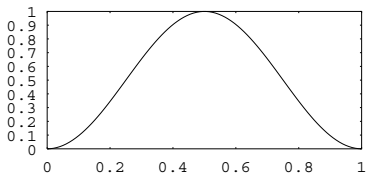


Figure 1.4: Transfer function of a bandpass filter constructed from the difference of a fourth-order and a second-order binomial filter.

1.5 Orientation-selective filters

The filters discussed in the preceding sections were frequency selective but (approximately) isotropic, i.e., only the frequency modulus $\omega = \sqrt{\omega_x^2 + \omega_y^2}$ was considered and not the phase. The class of *orientation-selective filters* can select frequencies according to phase *and* modulus. Most prominent in this class are the so-called *Gabor filters* which come in *quadrature pairs* and have filter masks that are derived from the continuous functions

$$\begin{aligned} o_x(\boldsymbol{x}) &= \frac{1}{2\pi\sigma^2}\cos(\boldsymbol{k}_0^T\boldsymbol{x})\exp(-\frac{\boldsymbol{x}}{2\sigma^2}) \\ o_y(\boldsymbol{x}) &= \frac{1}{2\pi\sigma^2}\sin(\boldsymbol{k}_0^T\boldsymbol{x})\exp(-\frac{\boldsymbol{x}}{2\sigma^2}) \ . \end{aligned} \tag{1.13}$$

The parameter $\boldsymbol{k_0}$ determines the modulus and phase of the frequency selectivity, whereas σ controls the width of the peaks in frequency space: large values of σ lead to narrow peaks and vice versa. Fig. 1.5 illustrates this relationship.

The term *quadrature pair* refers to the fact that o_y has a phase shift of $\pi/2$ with respect to o_y. The two filters are therefore selective for a common frequency modulus but orthogonal phases. The filter responses at a certain image point can be taken as the components of a two-dimensional vector, whose modulus is called the *oriented energy* at that image point. An example of the effects of Gabor filtering is given in fig. 1.7.

1.5.1 Gradient filters

In many applications, digital image processing techniques are desired that can reliably detect *boundaries* within an image, that is, discontinuities which may indicate the presence of an object. Sometimes this amounts to the detection of edges [11] or corners [48]; most of these and related techniques are based upon the evaluation of *gradient information* computed from an image. For this purpose, *gradient filters* are employed which are discretized versions of derivative operators. The simplest gradient filters read $G_x = G_y^T = (-1\ \ 0\ \ 1)$; examples of their effects are given in fig. 1.7. It is possible to construct larger gradient filter masks that are sensitive to image discontinuities at larger spatial scales [69].

Strictly speaking, gradient filters are orientation selective because they enhance horizontal or vertical structures. In many cases, however, the orientation

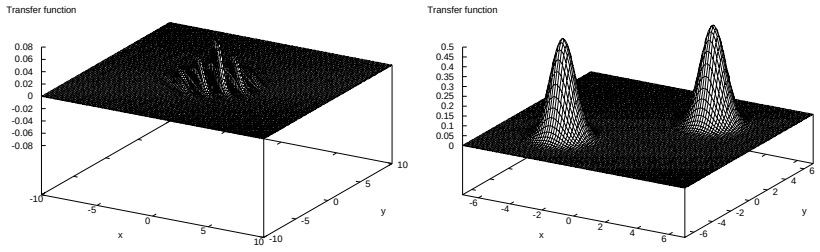


Figure 1.5: Left: Plot of the even part of the Gabor convolution filter. Right: Corresponding transfer function. For both diagrams, the parameters $k_{0x} = k_{0y} = 3,\ \sigma^2 = 2$ were used.

(phase) of local gradients is not needed but only their magnitude. Therefore, isotropic gradient filters would be desirable. Obviously, G_x and G_y are not isotropic, and it can even be shown [106] that there is *no* linear isotropic gradient filter. There is, however, a *nonlinear* gradient operator G that can be constructed from G_x and G_y. Let $F_S(x, y)$ again denote the sampled image; the definition of G then reads

$$(GF_S)(x, y) = \sqrt{(G_x * F_S)(x, y)^2 + (G_y * F_S)(x, y)^2}\ . \tag{1.14}$$

The analogous nonlinear operation that calculates the local phase is given by

$$(PF_S)(x, y) = \arctan\left\{\frac{(G_y * F_S)(x, y)}{(G_x * F_S)(x, y)}\right\}\ . \tag{1.15}$$

The nonlinear operator G is usually called the *gradient energy* operator. Often, the square root is not performed for reasons of computational efficiency. The rescaling of a gradient energy image to a fixed range of pixel values will therefore suppress small values strongly, leading to the smooth appearance shown in fig. 1.7. Linear gradient filters are a crude but nevertheless frequently used method of local orientation estimation.

1.6 Rescaling of digital images

Sometimes it is required to enlarge or shrink a digital image to a fixed size. In several chapters of this thesis, for example, neural classifiers is applied to each location in an image; however, since objects cannot be expected to have fixed size, and since the input dimension of the classifiers cannot be changed easily, one has to resize the image appropriately. In this section, the focus lies on

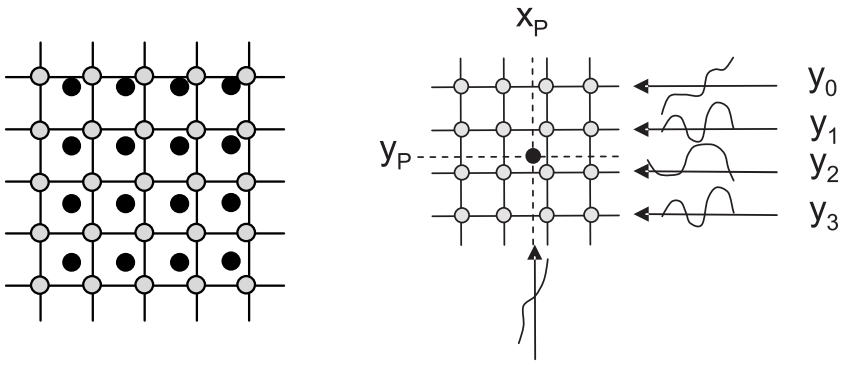


Figure 1.6: Deducing pixel value (x_P, y_P) by bicubic interpolation. Left: Grid positions of original and rescaled image are indicated by line cross-sections and black dots. Right: For obtaining the interpolated value at the non-integer grid position (x_P, y_P), a 4x4-neighborhood is taken into consideration. Four cubic polynomials (one for each row $0 \leq y_i \leq 3$) are fitted to match pixel values at integer grid positions $(\cdot, y_i)$. Now grid positions (x_k, y_i), which have non-integer x-coordinates x_k are available by evaluating the polynomials. Another cubic polynomial is fitted to match the four interpolated pixel values at each (x_P, y_i). By evaluating this polynomial at (x_P, y_P), the desired result is obtained.

the case where images have to be reduced in size: the two main techniques for achieving this are *interpolation* and *downsampling*. Interpolation is performed whenever an image is reduced in size by a non-integer factor; otherwise, the image can be shrunk to the desired size by downsampling (i.e., by sampling the image at integer pixel intervals).

When reducing an image in size, it is important not to ignore the sampling theorem. As was stated in section 1.1, this amounts to a reduction of the sampling frequency. Therefore, the highest possible frequency that can be represented correctly is reduced by the same factor. As a consequence, if the image is reduced in size by a factor of r, a fraction (given by $\frac{1-r}{r}$) of the highest frequencies must be removed from the image. If, for example, the dimensions of an image are reduced by a factor of 2, then the upper half of the spectrum (in both dimensions) must be removed by using appropriate smoothing filters. In this case, a suitable filter is a binomial filter of order 4 because it is separable, has a small filter mask and suppresses the upper half of the spectrum to very good approximation.

If $r \neq 2$, the smoothing operation must still be performed, although now with a smoothing filter that removes the necessary fraction of high frequencies. Gaussian filters are usually chosen for this purpose, and their variances are set accordingly. A more difficult issue is the fact that pixels in the resized image usually have non-integer positions in the original image. It is therefore unclear where the values of pixels in the resized image should be taken from, see fig. 1.6 for details. In this thesis, *bicubic interpolation* [75] of the smoothed original

image is used for this purpose. The idea is to locally approximate the image by fourth-order polynomials in both directions (this is why the method is called *bi*cubic interpolation) using a 4×4 neighborhood. The polynomials are then used to extrapolate values at non-integer grid positions. This process is illustrated in fig. 1.6.

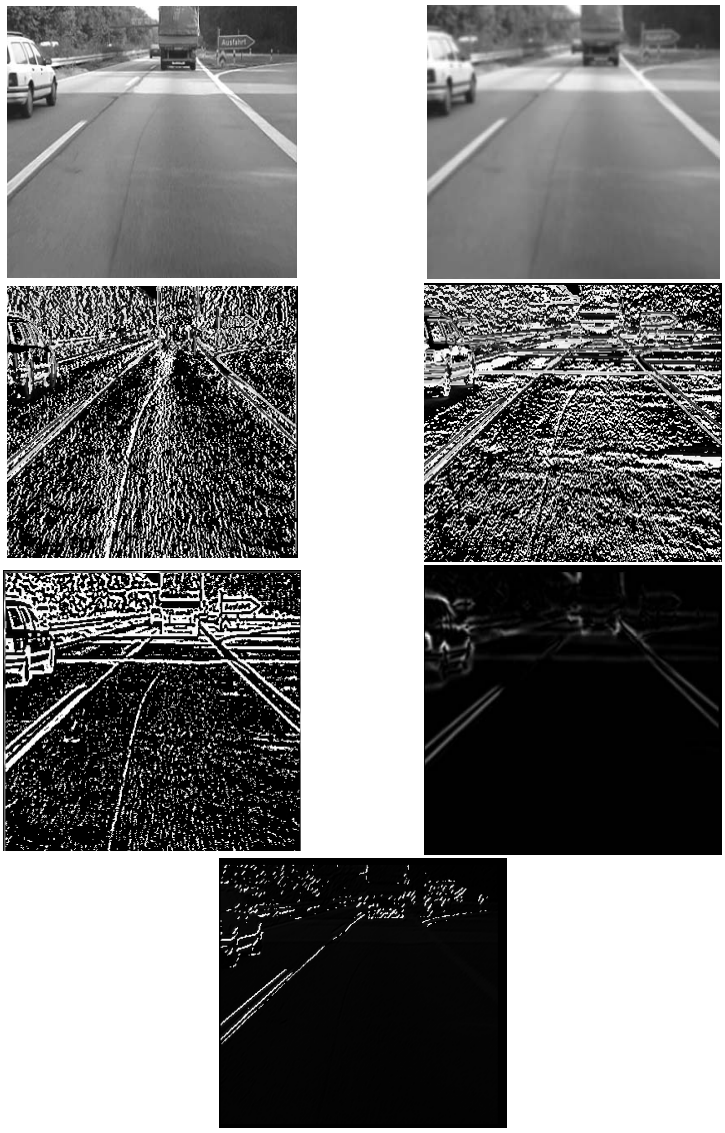


Figure 1.7: Examples of convolutions of an image (first row, left) with various filter masks. First row, right: Binomial smoothing filter of order 8. The filter mask is quite small, therefore only fine details are removed. Second row: Gradient filters G_x (left) and G_y (right). Third row: Bandpass filter constructed from two binomial filter of orders 4 and 8 (left) and nonlinear gradient energy operator (right). The image in the last row shows the application of a Gabor filter with parameters $k_{0x} = k_{0y} = 5$, $\sigma^2 = 1.5$. This means that orientations centered around $\pi/4$ are selected by the filter.

Chapter 2

Principal components analysis

Principal components analysis (PCA) is a method of statistical data analysis. It performs a linear transformation of the data to a often much lower-dimensional space while trying to preserve a maximum of information. Viewed from another perspective, it can also be interpreted as an unsupervised learning rule which has connections to the the well-known Hebbian learning rule [51, 88]. PCA has abundant applications in data compression [17], image analysis [117] and source separation [71]. In the context of computer vision and object recognition, PCA is often used for data reduction (see, e.g., [27], feature extraction (see [117] and references therein), allowing learning mechanisms to focus on significant properties of the data in a sense to be made more precise later. In this thesis, PCA is used to extend a neural network learning rule (see chapter 6) by an unsupervised component. This is done in order to extract the statistically most significant properties of the data, thus improving generalization ability. For this reason, the PCA technique is discussed in-depth with special emphasis on neural network implementations of PCA.

2.1 Notation

Given a set of vectors $\boldsymbol{x}_k$ of dimension K, we define notation for some important quantities: the expectation value of the $\boldsymbol{x}_k$ is denoted $E(\boldsymbol{x})$, the component-wise variance of the $\boldsymbol{x}_k$ is written $\mathrm{var}(\boldsymbol{x})$. In accordance with [57], the *correlation* of two components x^i and x^j of $\boldsymbol{x}$ is written $\mathrm{corr}(x^i x^j) = E(x^i x^j) - E(x^i)E(x^j) \equiv c_{ij}$. This quantity is explicitly defined because there is some confusion in the literature about it; some sources denote by correlation the quantity $\frac{E(x^i)E(x^j)}{\sqrt{\mathrm{var}(x^i)\mathrm{var}(x^j)}}$. The symmetric matrix $C = (c_{ij})$ is called the *correlation matrix*.

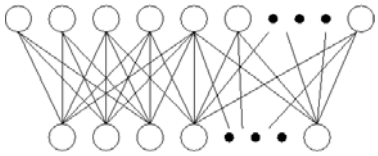


Figure 2.1: Illustration of the one-to-one correspondence between the PCA problem and feed-forward neural networks. The upper layer (the input layer) receives the vectors $\boldsymbol{x}_k$ one by one; connections from the input layer neurons to neuron m in the lower layer (corresponding to a principal component y^m) correspond to weight vector $\boldsymbol{w}_m$.

2.2 Basic concepts

PCA performs a linear transformation of each $\boldsymbol{x}_k$ to a vector of dimension $M \leq K$: $\boldsymbol{y}_k = W\boldsymbol{x}_k$ where W is a $M \times K$ - matrix whose rows are denoted $\boldsymbol{w_m},\ m = 0, \ldots, M-1$. Two constraints are imposed:

1. Each $\boldsymbol{w}_m$ must define, in descending order, directions along which the variance of the $\boldsymbol{x}_i$ is maximal: $\mathrm{var}(\boldsymbol{w}_m^T\boldsymbol{x}_i) = \max$.
2. The $\boldsymbol{w}_m$ must be mutually orthonormal: $\boldsymbol{w}_m^T\boldsymbol{w}_j = \delta_{mj},\ j \leq m$.

The elements y^i of $\boldsymbol{y}$ are called the *principal components* of $\boldsymbol{x}$. Note that the PCA problem can be translated into a two-layered feed-forward neural network with an first layer (containing K neurons) that receives the vectors $\boldsymbol{x}$ as input, and a second layer with $M \leq K$ neurons, each of which is fully connected to the first layer by the *weight vectors* $\boldsymbol{w}_m$. Please see fig. 2.1 for a visualization. The issue is now to find suitable $\boldsymbol{w_m}$ such that both constraints are fulfilled.

Before it is discussed how solutions to this problem can be obtained, some useful properties of the desired transformation are stated. The first point to be made is this: since the components of the $\boldsymbol{y}$ are sorted in direction of descending variance, PCA can be used to perform data compression by using only some (and not all) principal components. In practice, this is achieved by choosing $M < K$. When transforming back to the original space, a representation of the $\boldsymbol{x_i}$ is obtained which preserves most of the variability of the original vectors, although expressed by a lower-dimensional basis. Especially in image processing, it is highly surprising how few principal components are sufficient to represent an image with no perceptible change in visual appearance (please see fig. 2.2). This is possible because neighboring image pixels are usually strongly correlated. Since PCA is a linear operation, constraint 2 amounts to a decorrelation of principal components and thus to a removal of redundancies, making compression possible.

Indeed, and this is a second important point to be made, PCA can also be obtained by requiring a data compression property from the start: Consider the

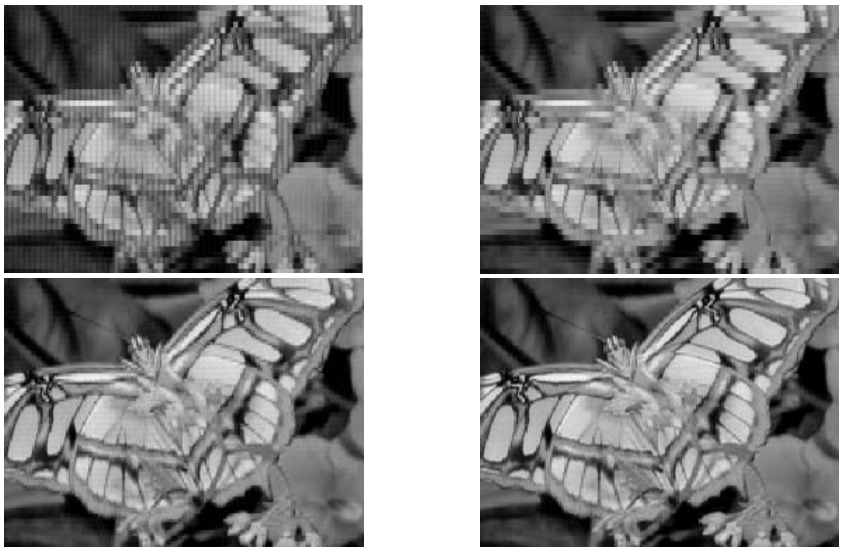


Figure 2.2: Examples of image compression and reconstruction by principal components. As in JPEG compression, the image is partitioned into non-overlapping blocks of 8x8 pixels, and PCA is performed using pixel values of the blocks as input vectors. If the PCA transformation is inverted, an approximation of the original image is obtained. The quality of the approximation depends on the number of principal components that were calculated; the pictures show image reconstruction using one (upper left), three (upper right), four (lower left) and five (lower right) principal components. At five principal components (using 5 instead of 64 numbers to encode one block), visual inspection cannot detect differences to the original image (not shown).

function

$$J_{MSE}(W) = E(||\boldsymbol{x} - \sum_{m=0}^{M-1} (\boldsymbol{w}_m^T \boldsymbol{x}) \boldsymbol{w}_m||^2) = \text{trace } C - \sum_{m=0}^{M-1} \boldsymbol{w}_m^T C \boldsymbol{w}_m \qquad (2.1)$$

which describes the approximation error in expressing the $\boldsymbol{x}_i$ by basis vectors from a lower-dimensional subspace. Solutions minimizing J_{MSE} can be shown [95] to be, up to a rotation, identical to the solutions defined by conditions 1 and 2.

2.3 Solution methods

It is well known [57] that the unit-length eigenvectors of the correlation matrix C fulfill both constraints. Thus, given the correlation matrix C which has to be estimated from available samples $\boldsymbol{x}_i$, it is possible to determine the principal components by standard numerical methods like the QR algorithm [23,44]. This

amounts to a *batch mode method* where the correlation matrix (i.e. the second-order statistical moments) is calculated first, followed by the determination of eigenvectors and eigenvalues. The disadvantage is obvious: the whole sample must be processed before a solution can be given. However, since this thesis deals with neural network learning, other methods are more interesting: first of all, an exact analytical solution is not required, but it is sufficient to approximate it with sufficient accuracy. This implies that a gradient-based algorithm might be used to find solutions by minimizing a cost function like J_{PCA} given in eqn. 2.1. Furthermore, it is unfeasible to process the whole data sample before solutions can be given; rather, a learning algorithm should consider individual vectors $\boldsymbol{x}$ one at a time and become increasingly accurate as more vectors are processed. This property is referred to as *online learning*.

2.4 PCA by neural networks

As shown in fig. 2.1, PCA can in principle be performed by feed-forward neural networks. It is only necessary to maximize the variance of $\boldsymbol{w}_m^T \boldsymbol{x}$ for all weight vectors by a suitable learning rule while enforcing the normalization and orthonormality constraints. It is, by the way, easy to see why normalization is required, since the variance of ${\boldsymbol{w_m}}^T \boldsymbol{x}$ grows linearly with the norm of the weight vector. The update rule for one weight vector can therefore be obtained by using gradient ascent. Considering that

$$\frac{\partial \mathrm{var}({\boldsymbol{w_m}}^T \boldsymbol{x})}{\partial w_m} = 2E(y^m x) = 2E((\boldsymbol{w}_m^T \boldsymbol{x})x \ , \tag{2.2}$$

we get the update rule

$$\boldsymbol{u}_m = \boldsymbol{w}_m + \gamma(y^m \boldsymbol{x}) \tag{2.3}$$

$$\boldsymbol{v}_m = \boldsymbol{u}_m - \sum_{j=0}^{m-1} ({\boldsymbol{w}_j^{(i+1)}}^T \boldsymbol{u}_m)\boldsymbol{w}_j^{\mathrm{new}} \tag{2.4}$$

$$\boldsymbol{w}_m^{\mathrm{new}} = \frac{v_m}{||\boldsymbol{v}_m||} \tag{2.5}$$

where γ is a small positive constant. If γ is chosen as the inverse of the sample size, the exact expectation value of eqn. (2.2) is recovered. This is, however, still not acceptable for a neural network implementation due to the orthonormalization steps (2.4) and (2.5).

2.5 Derivation of learning rules

A further simplification can be derived from the fact that γ is small. Therefore, the orthonormalization step can be linearized, mainly by multiplying out all terms in (2.4), neglecting all terms which are quadratic in γ, and then expanding the denominator on the right-hand side of (2.5). By expanding the square root in

powers of γ and disregarding terms of order higher than one, we finally obtain a learning rule known as the Stochastic Gradient Ascent rule (see, e.g., [91,92,94]):

$$\boldsymbol{w}_m^{\text{new}} = \boldsymbol{w}_m + \gamma y^m (\boldsymbol{x} - \boldsymbol{w}_m y^m - 2 \sum_{n<m} y^n \boldsymbol{w}_n) \ . \tag{2.6}$$

It has been rigorously proven that learning rule (2.6) achieves convergence of the $\boldsymbol{w}_m$ to the true eigenvectors of the correlation matrix under rather mild conditions (see [94]). The initially proposed version of this learning rule which is only suitable for the computation of the first principal component is usually called *Oja´s rule* [91]. Please observe that this is an online learning rule; it is therefore well suited for applications in neural networks (see fig. 2.1). Other learning rules have been proposed, most notably the *Generalized Hebbian Algorithm* [110] and the *Subspace Algorithm* [92]. They are similar in form to (2.6) and differ only in the term which achieves orthonormalization of weight vectors. Since the subspace rule will be important when generalizing PCA to the nonlinear case (see next section), it will be stated explicitly:

$$\boldsymbol{w}_m^{\text{new}} = \boldsymbol{w}_m + \gamma y^m (\boldsymbol{x} - \sum_{n=0}^{M-1} y^n \boldsymbol{w}_n) \ . \tag{2.7}$$

Unlike learning rule 2.6, the subspace rule is obtained by maximizing the sum of variances $J_{\text{SUB}} = \sum_{n=0}^{M-1} \text{var}(y_n)$ instead of the individual variances under the constraint of orthonormality. Therefore, this method gives the eigenvectors of the correlation matrix only up to a rotation (see [92] for a proof). This is the same type of behavior as obtained when minimizing J_{MSE} from (2.1). Indeed it is not hard to show that both functions are equivalent.

Due to its symmetry, the subspace rule can also be written in matrix notation:

$$W^{\text{new}} = W + \gamma \big\{ W \boldsymbol{x}\boldsymbol{x}^T - (W \boldsymbol{x}\boldsymbol{x}^T W^T) W \big\} \ . \tag{2.8}$$

The derivation of the subspace rule is equivalent in form and methods to the derivation of the Stochastic Gradient Ascent rule starting from eqns. (2.3) — (2.5). A curious and interesting fact is the appearance of the so-called *Hebbian term* on the right side of (2.6), (2.7) and virtually all proposed PCA learning rules, since it relates the well-known paradigm of Hebbian Learning [51, 88] to principal components analysis, suggesting that a processing principle similar to PCA might be employed in human and animal brains.

2.6 Nonlinear PCA

When using neural networks for PCA, the preceding sections implicitly assumed that the mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$ (or: from one network layer to the next) was linear, that is: $\boldsymbol{y}^i = \boldsymbol{w}_i^T \boldsymbol{x}$. Usually, however, this is not the case in current neural network models; instead, a *transfer function* of sigmoidal type is predominantly

used. The mapping thus changes to $\boldsymbol{y}^i = \sigma(\boldsymbol{w}_i^T \boldsymbol{x})$) where usually $\sigma(x) = \frac{x}{1+|x|}$ is used, although the main point is that $\sigma(x)$ is a nonlinear function. The question is now if and how the principles of PCA can be carried over to apply in this new situation. Numerous attempts in this direction have been made [50, 80, 93, 138] (or see [42] for a review) and the term "nonlinear PCA" is by no means unique. A basic problem in carrying PCA over to the nonlinear case is that the constraints that define the linear case do not define the nonlinear case uniquely. Therefore, it is not even clear how to formulate the task of nonlinear PCA. However, most authors have taken the minimization of the reconstruction error as given by the nonlinear version of (2.1) as the starting point of investigations. Therefore, a possible nonlinear PCA problem amounts to minimizing

$$J_{PCA}^{nl}(W) = E(||\boldsymbol{x} - \sum_{m=0}^{M-1} \sigma(\boldsymbol{w}_m^T \boldsymbol{x}) \boldsymbol{w}_m||^2) \tag{2.9}$$

under the constraint of orthonormality. Since it has been stated in the previous section that, for the linear case, this leads to the subspace learning rule, it seems like a good starting point to attempt a generalization of this learning rule to the nonlinear case. Indeed, a learning rule minimizing $J_{PCA}^{nl}(W)$ has been derived [72]; it is given here for completeness, for in the application presented at a later point of this thesis, an approximation is used: Nevertheless, switching to matrix formulation for clarity, the *nonlinear subspace rule* reads:

$$W^{\text{new}} = W + \gamma \big\{ F(W\boldsymbol{x}) W \boldsymbol{r} \boldsymbol{x}^T + \sigma(W\boldsymbol{x}) \boldsymbol{r}^T \big\} \ , \tag{2.10}$$

where $\boldsymbol{r} = \boldsymbol{x} - W^T \sigma(W\boldsymbol{x})$ represents the current approximation error (in the sense of (2.9)), $F(W\boldsymbol{x}) = \text{diag}(\sigma'(\boldsymbol{w}_0^T \boldsymbol{x}), \ldots, \sigma'(\boldsymbol{w}_{M-1}^T \boldsymbol{x}))$ and $\sigma(W\boldsymbol{x}) = (\sigma(\boldsymbol{w}_0^T \boldsymbol{x}), \ldots, \sigma(\boldsymbol{w}_{M-1}^T \boldsymbol{x}))$. In [71] it is shown that the first term in (2.10) can be neglected under certain conditions, and furthermore that the nonlinear subspace rule can be consistently approximated by

$$W^{\text{new}} = W + \gamma \sigma(W\boldsymbol{x}) \big\{ \boldsymbol{x}^T - \sigma(W\boldsymbol{x}) W \big\} \tag{2.11}$$

in the sense that this rule also minimizes J_{PCA}^{nl} from (2.9). Now, this is an extremely interesting result since by taking $\sigma(x)$ to be the identity map one recovers the original subspace rule (2.7). This result therefore suggests that one can perform nonlinear PCA in a neural network in the same way as linear PCA using the subspace rule. It is just required to replace occurrences of $\boldsymbol{w}_m^T \boldsymbol{x}$ in the learning rule by $\sigma(\boldsymbol{w}_m^T \boldsymbol{x})$, which usually happens automatically when changing to $\sigma(x)$ as a transfer function.

2.7 Local PCA

The PCA setting elaborated up to now can be termed "global" in the sense that all computed principal components are principal components of the *whole* input vectors. To put it another way, each second-layer neuron in fig. 2.1 is connected

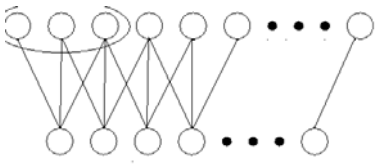


Figure 2.3: Example of a feed-forward neural network that performs local PCA. The ellipse indicates the set of neurons projecting to the leftmost second-layer neuron.

to *all* input neurons. This contradicts findings from neurophysiology where it has long been known that neurons in the early stages of visual processing have *local connectivity*, i.e., they are connected to a small patch of the preceding layer only [28]. The concept of *local feature analysis* [101] tries to reflect this property by representing images as a superposition of spatially localized properties (see also [97] for a motivation of local image features). The concepts of linear as well as nonlinear PCA can be used without restrictions for local feature analysis; the term *local PCA* has become common for such algorithms.

When performing local PCA by a neural network, one only needs change the full connectivity of fig. 2.1 to local connectivity: see fig. 2.3. In chapter 6, local PCA is performed by using a nonlinear subspace rule and local connectivity similar to fig. 2.3.

Chapter 3

Object detection: selected approaches

The goal of this thesis has been to develop neural learning methods that can improve object detection systems. There is a great deal of research currently going on in this field, and various methods exist to tackle the problem for a range of application scenarios. In this chapter, selected proposals and ideas that have been sources of inspiration are reviewed and discussed. The discussion is restricted to models who intend to be of general applicability in real-world scenarios; therefore, specialized solutions making restrictive assumptions about the types of objects and scenes that can be expected are not considered: this excludes, for example, methods often used in robotics where fixed backgrounds and a limited number of objects can be expected. Many of the models described in the following are (to various degrees) inspired by biological visual processing, although in some cases the relationship is rather remote, and real-time aspects are seen as more important. The following overview is not intended to be complete; nevertheless, each of the presented models has one or several aspects which are addressed and optimized by work presented in this thesis.

3.1 Saliency-based object detectors

Several proposals attempt to model the stimulus-driven allocation of visual attention by so-called *saliency maps*. These are two-dimensional representations computed from an input image exhibiting high values at locations that attention should be directed to. Effectively, "interesting" image regions are highlighted and "irrelevant" image regions are disregarded by saliency maps. In order to reliably detect conspicuous image locations, these models operate in a purely feature-based fashion on the whole input image. They typically compute features like color, orientation, gradient energy or similar features at each image location. The idea behind computing many different local features is the observation that most objects have one or more local properties which "stand out"

from the surrounding image background, thus making efficient search possible. This approach, however, works only if a sufficient number of local features is computed.

Another assumption concerns the issue what local properties should be treated as "conspicuous". Saliency map models assume that locations which differ strongly from their surroundings merit the allocation of attention. This naturally leads to the concept of center-surround filtering. As discussed in chapter 1, center-surround filters (also called difference-of-Gaussian or doG filters) are non-separable bandpass filters which analyze an image at one spatial scale. This makes it necessary to evaluate local conspicuity at several spatial scales. The predominantly used technique is that of *image pyramids*, which downsample the image several times by a chosen factor (see chapter 1) and use the saliency map to perform the detection of conspicuous locations on each downsampled image.

A further crucial point in computing stimulus-driven saliency is *competition*. In accordance with psychophysical findings [134], competition is modeled as occurring between neighboring (local competition) or all (global competition) conspicuous image regions. Competition leads to the suppression of sub-maximal map activations and ensures a unique "winning" location within the radius of competition.

From the strongest local activation in the map, a *focus of attention* (FOA) can be determined. This location is considered the most conspicuous, and thus the saliency map recommends it for further processing, possibly by a recognition system. Subsequently, the FOA is inhibited (this is termed *inhibition of return* in the literature, see, e.g., [76]) and the competition mechanism is applied again. This will lead (due to continuing input from the image) to the emergence of local activations that were previously suppressed, and a new FOA can be determined. By repeating this procedure, the most salient regions in an image (in decreasing order) can be visited by the FOA and thus be recommended for further analysis.

In this section, a saliency map proposed in [65] is described as a prototype of a purely stimulus-driven bottom-up object detection system. A sketch of the saliency map architecture is given in fig. 3.1. It analyzes the image at 8 spatial scales using three *modalities*: local orientation, local intensity and local color. For each modality, a number of *feature maps* is created, representing the first stage of processing and corresponding, e.g., in the case of the modality for local orientation, to the input image linearly filtered by orientation-selective filters. Within each modality, a number of doG-operations is applied to calculate local conspicuity, combining feature maps of different scale and type but always within the same modality. The results of the second processing step are summed up to form, for each modality, a single *conspicuity map*.

The final saliency map is obtained by a linear combination of all conspicuity maps. An inhibition of return mechanism is implemented using mechanisms described above.

The model is tested on synthetic images of a type often used in visual search experiments, and it is demonstrated to reproduce two typical kinds of search slopes, namely those attributed to "serial" and "parallel" searches (although

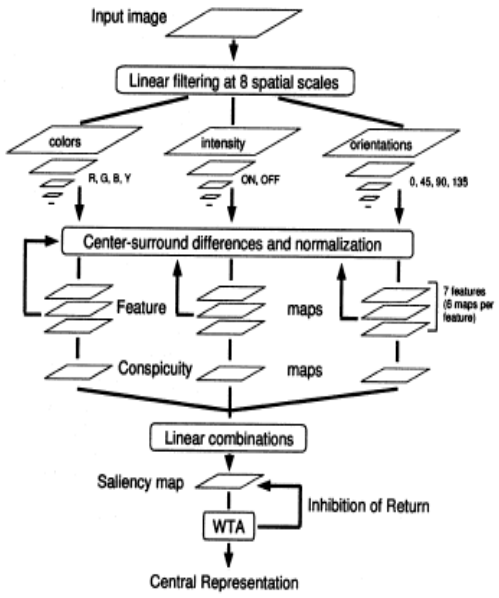


Figure 3.1: The saliency map architecture proposed in [65].

current consensus is that these terms are misleading).

Furthermore, tests are performed using real-world images of outdoor scenes where the task is to detect military vehicles. The model is very successful in this task, but should be clear that this can only be the case because the target objects size matches the size of the FOA which is set accordingly. No object model is used at all, only the implicit assumptions about what constitutes a salient image location.

In chapter 5, ways are described to incorporate the learning of simple feature-based object models into a model very similar to [65]

3.2 Object recognition and classification

Once an object hypothesis has been produced by a saliency map or another method, it is the task of an object classifier to analyze that hypothesis further in order to confirm or to reject it. Since the number of proposals which have been put forward to solve this task in general or specialized settings is enormous, only general principles to be found in most classification models are outlined, along with a few representative proposals.

The input to most classification models is a (rectangular) image region (often termed *region of interest* – ROI) which is to be classified. In most cases, classification is preceded by a *feature extraction* step, during which significant visual properties from within the given ROI are calculated. The result of feature

extraction is then passed to a trainable classifier such as a support vector machine (SVM), a neural network (NN) or a decision tree [121]. Consensus which method yields superior performance has not yet been reached, therefore all three models (and others) are widely used for recognition and classification. Likewise, the correct choice of feature extraction for a given classifier and classification task is a controversial issue. It is known that the performance of classifiers (e.g., support vector machines) can be significantly impaired if too many "irrelevant", that is, uninformative, features are extracted [46].

Independently of the choice of classifier, it is also considered to be the task of successful feature extraction to increase the robustness, i.e., the invariance of the classification to distortions or noise, thus effectively improving the generalization ability of the whole system. Typical distortions are due to lighting changes, small translations, rotations or scalings or simply due to noise.

Popular feature extraction methods are the scale-invariant feature transform (SIFT) [85], Gabor wavelets [98], principal components (see chapter 2), Haar wavelets [69], integral image features [121], histogram features [105] and corner features [48]. These algorithms can be used separately or in combination with each other to obtain robust and powerful classifications.

In chapter 4 of this thesis, a combination of fixed feature extraction and trainable neural network classifier is presented in the context of a car detection task which uses a manually designed method to detect car hypotheses. In chapter 7, the same neural network classifier is simultaneously optimized w.r.t. the (often conflicting) objectives of speed and accuracy. In chapter 6, an approach is presented which unifies initial detection, feature extraction and neural network classification.

3.3 Combinations of object classifiers and saliency-based detectors

There are several models which try to combine the detection of ROIs by saliency maps coupled with subsequent ROI classification. In [124], the saliency map model of [65] is combined with an object classifier described in [105]. A similar model using a slightly more advanced saliency map and a robust real-time object classifier [121] is presented in [34]. It is claimed in [34] that the saliency map can be supplied with trainable feature cues in order to detect different kinds of objects but the training method is not described.

Both publications report a significant reduction of the image area that effectively needs to be processed by the object classifiers. This effect was also observed during the investigations described in chapter 5 of this thesis. Additionally, chapter 5 presents a concept of learning to enhance object features and to suppress non-object features.

3.4 Whole-image search using object classifiers

Although it seems an infeasible approach at first glance, it is conceivable to apply a fixed-size object classifier at each possible and distinct image location (at several spatial scales) in order to perform both object detection and classification. If this method is to work, very fast and efficient classifiers are required, and indeed most classification methods are not suitable for this kind of whole-image search (also commonly termed "brute-force search"). The class of *convolutional neural networks* (CNNs) [83] is a type of neural network classifier permitting whole-image search due to a network model that can be implemented by successive convolutions of the input image (see also chapter 1). Fig. 3.2 gives a (slightly simplified with regard to [83]) sketch of the CNN model. An important property used by CNNs is the concept of *weight sharing*. This means that identical weight configurations are duplicated over all spatial locations within one processing layer of the NN. This amounts to a convolution (see chapter 1 for an overview) of a layer's two-dimensional activations with a linear filter defined by the NN weights. Although CNNs are primarily designed for very fast execution speed, they exhibit a fair amount of biologically realistic properties, namely local connectivity and a gradual change from topographic to categorical representation. The latter property is ensured by the repeated subsampling operations, which reduce the spatial accuracy of the representation in favor of an increasingly global representation of whole objects in the lower layer of a CNN. Using CNNs, it has been demonstrated that brute-force object searches can be performed in real-time if necessary. Since CNNs are fixed-size classifiers, brute-force search has to be conducted at several spatial scales (see chapter 1 for details).

The real-time capability is demonstrated in [83] and more recently in [37], where a real-time face detection system is presented.

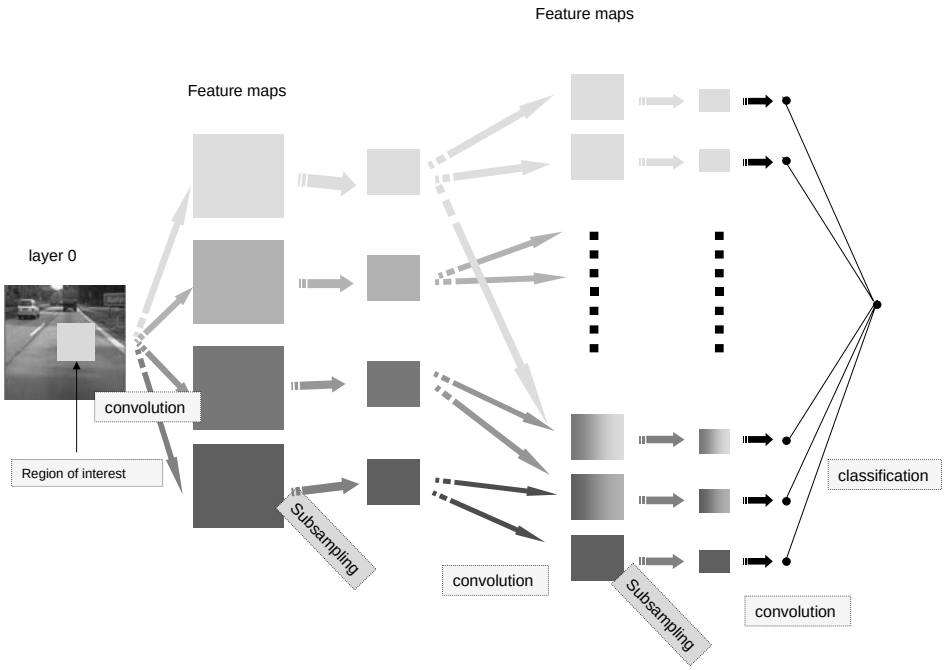


Figure 3.2: The CNN architecture proposed in [37]. The rectangle in the input image indicates the ROI to which the CNN is applied. Convolution filter sizes are chosen to be small (5 pixels), and a few manually chosen convolution results (feature maps) are summed up before downsampling. This is indicated by more than one arrow ending at a feature map. Subsampling decreases the size of a feature map by a factor of 2. In the last convolution operation, each feature map (already quite small due to repeated subsampling) is "convolved" with a filter that has the same size as the feature map, producing a single number. This CNN architecture can also be used for brute-force searches: feature maps are then obtained by convolving the whole image (instead of an ROI) with the trained filters.

Part II

Own work

Chapter 4

Feature design for object classification and detection[1]

4.1 Summary

This chapter is mainly concerned with object classification and the design of meaningful visual features. Based on the problem of car detection, a suitable feature extraction is defined, its invariance properties are tested and a working neural network classifier is presented which relies on the extracted features. It is demonstrated how a simple network optimization method can improve the speed of classification, and how the feature extraction and the neural object classifier can be embedded into a real-time car detection system.

[1]Some of the content of this chapter has been published in: A. Gepperth, J. Edelbrunner, and T. Bücher. Real-time detection and classification of cars in video sequences. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 625–631, 2005.

4.2 Introduction

In many applications in driver assistance systems, behaviorally relevant objects in traffic scenes must be reliably detected by image processing in order to generate high-level-representations allowing, e.g., behavior planning. Most prominent among behaviorally relevant objects are certainly cars and pedestrians. Real-world problems such as, for example, visual pedestrian or car detection are considered to be very tough problems since objects can exhibit a very large amount of variability. Reasons include viewpoint dependency, intrinsic within-class variability, object occlusions and image transformations due to lighting changes or insufficient sensor performance. On the other hand, there are also advantages that can be exploited, namely the adherence to basic physical laws that is guaranteed, which makes sure that relevant objects behave in a predictable way.

For cars and pedestrians there exist numerous proposals for detection and confirmation strategies. A perfectly accurate and universally reliable object detection system either for pedestrians or cars remains, however, elusive. In the following paragraph, some interesting developments in these domains are reviewed, concentrating on car and pedestrian classification rather than detection since classification is the focus of the work presented here. Given the vast amount of research which is being done in this domain, however, this is not intended to be either a complete or a representative list, but merely a selection of interesting approaches which serves to highlight the intrinsic properties of the presented approach in contrast or accordance to those that are mentioned here.

The most basic property that distinguishes different proposals from each other is the choice of features upon which the classification is based. In analogy to human and primate vision, classifications are performed using stereo information [31] as well as monocular image features such as the outline (shape) of objects [2], wavelet coefficients extracted by linear filtering [99, 112] or Haar wavelet decomposition [121], principal component analysis [117] and local orientation coding techniques [40, 114], to name just a few. There are also systematic approaches to automatically select an optimal subset of image features from a given feature base using either evolutionary algorithms [117] or boosting methods [121]. Another source of diversity are the methods employed to reach a classification decision: typical methods are template matching [99], neural networks of various architectures [43, 133] and support vector machines [117].

Driver assistance systems typically operate under real-time constraints, although high detection accuracy is nevertheless crucial. In this chapter, a method to reliably and efficiently detect cars within the driver assistance framework developed at our research group is described. The framework consists of a number of independent but interacting modules each of which performs a specialized analysis task on a monocular, gray-valued video sequence that is common input to all modules. The classification is based on local orientation coding derived from local edge information. The feature base was designed, i.e. not constructed using an automated procedure as mentioned previously, and the classification is done by an artificial neural network which learns from examples in a supervised

way. An initial detection module for vehicles was already in existence within the general framework; goal of the work presented here is to show how hypotheses produced by the initial detection module ("regions of interest" - ROIs) can be evaluated (i.e. classified) by a confirmation module.

4.3 System architecture

Advanced driver assistance applications perform either safety oriented functions such as lane departure warning, lane change warning and pedestrian detection, or comfort oriented functions, e.g., traffic sign detection. Among others, specialized modules have been implemented which carry out all of these tasks. Common input to all modules are preprocessed data obtained from the input video image; by using a common feature basis, it is ensured that the image need only be preprocessed once per frame, thereby facilitating real-time application.

In order to make the car detection fast enough, the process of finding cars in image sequences is hierarchically organized. The different modules used for car detection are initial detection (different algorithms provide vehicle hypotheses, i.e. ROIs), confirmation (scale invariant evaluation of detected ROIs) and tracking (localizing a given ROI in the following video frame). For a detailed description of the tracking module, see [10, 55] and references therein. Obviously, the results of the different modules (except the feature extraction) are not independent and therefore a temporal coupling structure is implemented to increase the reliability of the car detection results. All specialized modules use features calculated in the preprocessing step. In spite of the complexity of the task, the initial detection guarantees fast data reduction such that only a fraction of an image actually needs to be analyzed thoroughly.

The ROIs generated by the initial detection module are presented to the confirmation module which generates a conspicuity measure for each ROI indicating how likely it judges the ROI to contain a car. The number of initial hypotheses is thereby reduced depending on the "strictness" (which effectively expresses the minimum conspicuity that will still be interpreted as a "car present" decision) of the confirmation module. This property is governed by a single threshold value which must be set according to the desired results. The remaining ROIs are tracked; any incorrect hypotheses must be eliminated by heuristics governing the interaction of the initial detection, confirmation and tracking modules: The tracking module output is continuously compared to hypotheses generated by the initial detection modules. A confidence value is maintained for each tracked ROI: the confidence is incremented if the tracked object coincides with a confirmed hypothesis, and decremented if it does not. When the confidence decays below a certain threshold, tracking is turned off for this particular ROI. It is then assumed that the hypothesis has become invalid or has been invalid all along. Thus, incorrectly confirmed hypotheses can be eliminated. This heuristic takes several frames to discard incorrect hypotheses, besides the fact that tracking is a computationally expensive procedure. Therefore, it can make sense to use a more powerful classification to save effort later. To summarize, the use

of a confirmation module yields the advantage of increased speed due to fewer false detections on the one hand, and on the other hand it provides an effective way to obtain an independent quality measure of tracked ROIs, which can be used to calculate more accurate confidence values for each tracked ROI.

4.4 Image analysis and feature selection

The extraction of meaningful object features is an important issue for any image processing algorithm. In contrast to the first driver assistance applications on the market (e.g. lane-departure-warning systems) where the preprocessing stage was directly linked to the application-specific processing algorithms, the car detection and classification methods described in this chapter are implemented as modules which are embedded into a larger driver assistance application.

4.4.1 Preprocessing

For such system architectures, the proposal is to calculate a common high-level feature basis which can be accessed by all image processing algorithms. This approach has a number of advantages: First of all, the development of robust task-specific algorithms is simplified significantly; secondly, assuming that the feature basis has certain invariance properties e.g. with respect to varying lighting conditions, such properties will be automatically incorporated in the task specific algorithms. And thirdly, depending on the number of modules, even the whole processing time can be reduced due to simplified processing in additional task-specific image processing algorithms. Furthermore, the calculation of the feature basis can be implemented on dedicated hardware, so that a general purpose processing core can be used for the succeeding algorithms.

The chosen feature base consists of horizontal gradients, vertical gradients, gradient energy, contour points including a quantized local orientation, and line segments including the mean energy along each segment. The features up to the contour points are represented in terms of images and are calculated from gradient filter results (see chapter 1) and the Canny edge detector [11]. For efficient access of sparse contour points, a linked data structure is used which is established during the non-maximum suppression algorithm of the Canny-Filter. The line-segments are obtained by a clustering algorithm that makes use of this linked representation; the involved calculations are not presented here as they are beyond the scope of this chapter (but see [10]). Each line-segment is represented by the image coordinates of its end-points and the mean gradient energy along that line and therefore provides an extremely sparse coding of an image contour. For typical road scenarios the average number of line-segments obtained varies between 250 and 500 (image size: 496x256, minimal line-length: 5 pixels).

4.4.2 Features for Classification

To allow for a real-time classification of a given ROI, the data provided by the preprocessing must be processed further, while still providing enough information for a reliable decision. The outcome of this process shall be termed a *feature set* in what follows. A sub-optimal choice of extraction procedure can significantly reduce classification performance; because of this, some effort was made to identify suitable features. An upper boundary on the complexity of the feature extraction is set by the required calculation time, since real-time decisions are desired. This excludes common methods in image processing like Gabor or Fourier transforms (see chapter 1), which need not be a disadvantage provided it is possible to identify cars using only "primitive" features. By no means it is assumed or concluded that this is true for more general object recognition applications.

In order to make object recognition invariant with respect to scaling, it is demanded that this invariance should be already incorporated into the feature extraction process.

Histograms

In the course of investigations, an extraction method was identified which is equally favorable in terms of processing time as well as suitability for classification. This method is referred to as the *Set of Orientation Energies* or SOE method. It will be described in detail further below.

First of all, the ROI is subdivided into a fixed number of rectangular regions called *receptive fields* in analogy to biological image processing. From each receptive field a set of numbers is extracted by the chosen feature extraction scheme. The concatenated set of numbers extracted from all receptive fields in an ROI represents the feature set corresponding to that ROI. The point is that the number of receptive fields does not depend on ROI size: the larger the ROI, the larger the receptive fields. This incorporates already at a fundamental level the requirement of scale invariance: the feature set does not reveal the size of the ROI it was calculated from. It is evident that a finer subdivision of an ROI gives higher spatial resolution and therefore more precise spatial information; on the other hand, it leads to greater processing costs and larger feature sets, which in turn slow down the classification process. Determination of the optimal subdivision was achieved by varying the number of receptive fields in each direction. Starting from 1×1, it was increased until classification results stopped improving.

The SOE method simply computes the sum of energies of all identically oriented edges in a receptive field and normalizes this sum by the total energy of edges within the receptive field. Orientations are quantized, that is, ranges of angles are mapped onto integers, thus effectively reducing the number of possible orientations. The output of the algorithm is a number for each orientation, indicating the ratio between oriented energies of that particular orientation and the total (summed-up) energy of all orientations, and is therefore a number

between 0 and 1. By analyzing the problem class and doing some experiments, it could be verified that four orientations are sufficient; the algorithm therefore produces 4 numbers per receptive field.

Invariance properties

To assess the invariance properties of SOE, features from ROIs in natural as well as artificial video sequences and from respective transformed versions are computed using a 7x7 subdivision of the ROI. The transfo¡rmations are scaling (of ROI and image) and translation (of the ROI). Results are measured on ROIs containing vehicles/objects in two video sequences. One sequence is synthetic, the other recorded on a highway. Results are averaged over all frames of each sequence. For the investigation of translation invariance, ROIs are shifted to

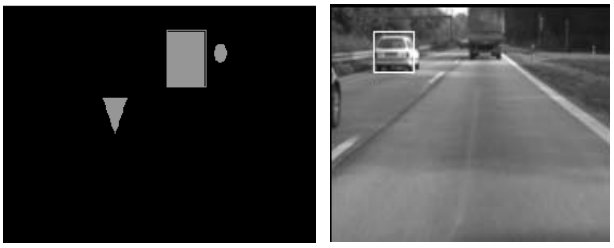


Figure 4.1: Example images from synthetic and real-world video sequences used for the testing of invariances. Typical objects that were used for feature extraction are in boxes.

the left and to the right by fixed percentages of their width. Without loss of generality, only left/right shifts are considered since the SOE method does not prefer certain directions. For testing scale invariance, the image is downsampled and smoothed by appropriate linear filters (see chapter 1); ROI dimensions are reduced by the same factor. Two error measures are defined describing the deviation between a feature set and its transform. One is essentially the mean squared error (MSE) normalized by the average of the original feature set, but correcting for the fact that the 4 numbers that are generated per receptive field are not independent (they must add up to 1.0). Therefore, the MSE is halved; this method is referred to as *corrected MSE*. The other measure (referred to as *binary deviation*) takes the relative ordering of the orientations into account. First, it transforms the feature sets to be compared into binary feature sets by substituting a value of 1.0 if an element is among the two (absolutely) strongest within a RF, and 0 if it is not. The MSE between the binary feature sets is given back as an error measure, producing a number between 0.0 and 1.0. The results on synthetic images of rectangles and triangles on dark background are shown in fig. 4.2. Note that the method is almost invariant to translation until 20% of ROI width. The reason is that, with a 7x7 receptive field configuration, one receptive field corresponds to 28% of ROI width. As long as critical features stay

within the same receptive fields, no changes can occur. Notable is furthermore the low error introduced by scaling.

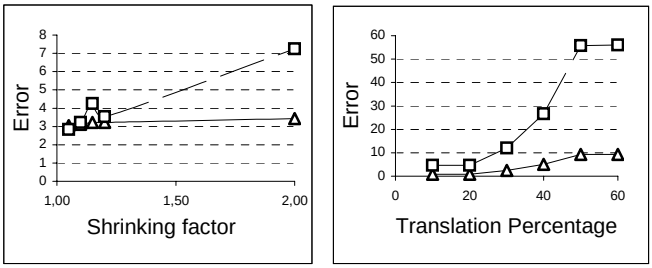


Figure 4.2: Errors introduced into a feature set by scaling (left) and translation (right). Measurement was done on synthetic images Shown is the binary deviation (triangles) and the corrected MSE (squares). Both measures are explained in the text.

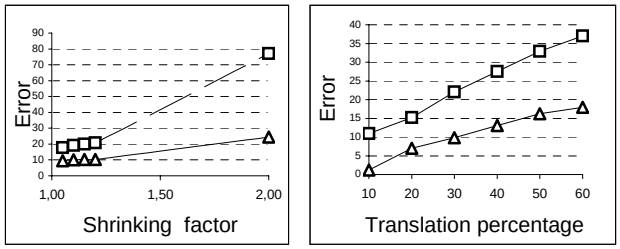


Figure 4.3: Errors introduced into a feature set by scaling (left) and translation (right). Measurement was done on real-world images. Shown is the binary deviation (triangles) and the corrected MSE (squares).

Results on real-world images (fig. 4.3) are poorer, as might be expected since translation introduces new and unpredictable content into an ROI instead of empty background. At first glance the SOE method seems to perform poorly under scaling in particular, but at second glance one can perceive that the structure of a feature set remains quite unaffected. This becomes apparent when considering the binary deviation measure. which changes by only 15% under 50% scaling; this may well be the reason why classification performance is quite independent of ROI size (see later sections, especially fig. 4.5).

4.5 Generation of hypotheses

The generation of vehicle hypotheses is the first processing stage within an architecture for car detection. In this section, the algorithms generating the ROIs presented to the confirmation module are briefly discussed.

Two different algorithms for generating initial vehicle hypotheses were developed which are based on different image features. One algorithm employs the line-segments calculated in the preprocessing stage for producing a list of potential vehicle positions: The middle of each approximately horizontal line-segment serves as a starting position for searching lateral vehicle boundaries. Based on the location of the line-segment in the image, the image region occupied by a vehicle at that location is estimated. In this region a one-dimensional signal is calculated by vertically projecting horizontal gradient information. Extraction of local maxima, analysis of maxima positions and signal values finally leads to either acceptance or rejection of that region.

The other strategy for calculating potential vehicle positions utilizes estimates of lane borders provided by another module and is therefore based on higher-level knowledge. It evaluates a robust measure of mean gray-value in each row (up to a maximal predefined distance from the ego-vehicle) for each lane, resulting in hypotheses for vertical vehicle positions for each detected lane. In order to suppress false detections due to horizontal shadows, the same mechanism for evaluating the presence of horizontal vehicle boundaries as in the previously sketched algorithm is employed.

4.6 Requirements a confirmation module

For the implementation of the confirmation module, the following properties are required:

- adaptivity: can be trained by examples
- flexibility: able to generalize when dealing with previously unseen data
- good performance: capable of real-time decisions
- robustness: up to a degree, invariant to scale, translation and a wide range of lighting conditions.
- independence: should not rely on other modules

In the light of the explanations given so far, it is evident that the last three constraints are fulfilled by construction: the performance constraint is taken care of by the way input features for the confirmation module are generated, provided only that the confirmation itself can be implemented efficiently (which shall be shown later). The robustness constraint is satisfied on the one hand by using the results of the preprocessing stage which already exhibit a great deal of invariance properties, and on the other hand by the invariance properties of the feature sets extracted from each ROI which serve as inputs to the confirmation module. Lastly, the independence property is ensured by the fact that only the input feature sets determine the decision of the confirmation module. The first two constraints are satisfied by the implementation of the classification function within the confirmation module which will be described in the next section.

4.7 The confirmation module

The previously stated requirements on the confirmation module do not specify a particular implementation, and indeed several alternatives present themselves, most prominently multilayer perceptrons (MLPs) and support vector machines (SVMs). For now, issues that are independent of this particular choice are discussed.

4.7.1 Training data

The confirmation module is required to decide whether an ROI that is presented contains a car or not, therefore distinguishing two classes of objects (cars and noncars) from each other. Training examples consisting of an ROI and a class label (0.0 or 1.0) are generated from a significant number of different videos of typical highway scenes. Four disjunct datasets termed D_{train}, D_{val}, D_{test} and D_{ext} are created, each containing 5000 examples, which are randomly taken from a larger database of over 100000 examples. 50% of the examples in each set are positive examples. The positive examples are labeled manually, the negative examples are produced using the initial detection module: for each frame of a video sequence, its output (a collection of ROIs) is compared to the previously labeled positive examples. All ROIs that do not coincide (within a certain tolerance range) are added to the database as negative examples. In this way, it is ensured that the negative examples used for training are comparable to those encountered during application. Furthermore, if the initial detection module should be altered, negative examples can simply be generated automatically by using the new initial detection module.

The positive examples must fulfill a number of specifications. Positive examples must have a certain width/height ratio (close to 1 for cars), contain only cars and no trucks; they must include the pronounced lower edge of cars, enclose all of the rear view of a car tightly and vary over all scales. The last condition is a precaution: although the feature sets are roughly invariant to scaling, it is prudent to account for small deviations by including training examples of all sizes. For negative examples, all conditions are fulfilled by construction.

4.7.2 Decision making

One expects the output of the confirmation module to be a continuum of values between the labels for the two classes. In order to interpret this as a decision in favor of a particular class, a threshold is defined, and the decision for a particular class is then expressed by a module output that exceeds or does not exceed the chosen threshold. The underlying assumption that justifies this simple method is that the degree of match between objects and learned models is at least approximately encoded by the euclidean distance between the module output and the corresponding class label. If this assumption holds, the module outputs can be expected to be strongly centered around the class labels (in this case, a strongly bimodal distribution is obtained), and an optimal threshold can be

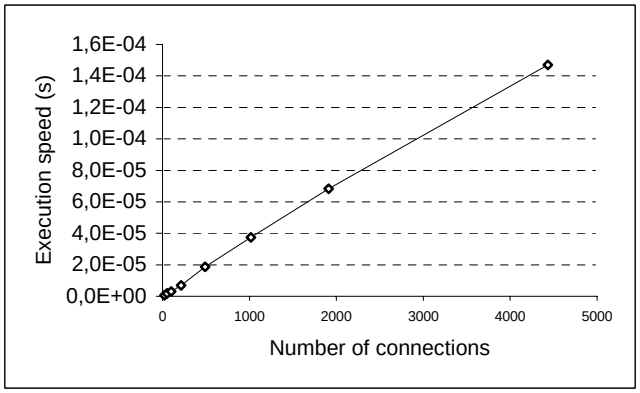


Figure 4.4: Dependence of execution speed on NN complexity.

applied that separates the outputs with minimal error. This assumption needs to be checked explicitly.

Implementation of the classification function

For the implementation of the classifier, a MLP is used which converges onto a single output neuron. This choice is made because better classification speed can be achieved by NNs than by SVMs while obtaining comparable classification accuracies. One hidden layer is used, and all activation functions are of logistic sigmoidal type. For training, a modified form of the Rprop learning algorithm [62] is used. The layers are fully connected, and all neurons in the input layer are origins of shortcuts (connections that bypass one or more layers) to the output neuron.

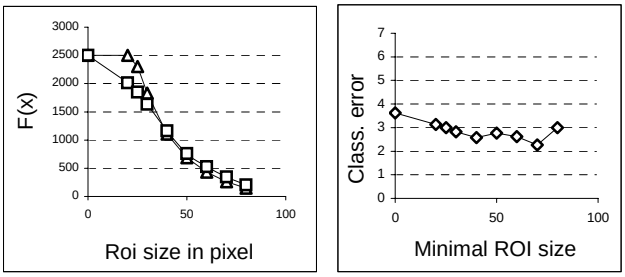


Figure 4.5: Dependence of classification accuracy on ROI size. Left: size distribution of examples in D_{test}. Car ROIs are depicted by squares, non-car ROIs by triangles. F(x) denotes the cumulative distribution function, i.e. the number of ROIs wider than x. Right: classification accuracy plotted against the lower bound on ROI width.

Initially, a population of NNs with different randomly drawn, small weight

values is generated. Each member of the population is subjected to an iterated optimization loop, a step of which consists of NN training and subsequent magnitude-based pruning [103]. This simple pruning heuristic was chosen because initial attempts using a sensitivity-based method yielded poorer results at higher computational cost and were therefore abandoned. For a review of pruning techniques, please see [103]. The goal of this procedure is to obtain a NN that has as few connections as possible (because execution speed scales linearly with the number of connections in the used implementation, see fig. 4.5) while still being capable of the best possible classification. Motivated by general statements about the learning capacity of MLPs [103], the unoptimized NNs have about 5000 connections. Since it is a difficult issue to show analytically which number and size of hidden layers is optimal for a specific problem [4, 103], no attempt is made to tackle it; it is left to an improved optimization technique. For this purpose, evolutionary NN optimization methods seem the methods of choice. For a comparision of magnitude-based pruning to an evolutionary NN optimization method combining the abilities to reduce and increase the size and complexity of NNs, see chapter 7.

Inputs to the NN are the feature sets that are generated in the previous processing step of feature extraction, so the input layer must have the same size as the feature sets which is 196. For the implementation of the NN, the ReClaM package of the freely available SHARK library (http://shark-project.sourceforge.net) is used.

NN Training

NN training is embedded into the optimization loop described above. For training, the MSE is used as an error measure. As usual, weights are initialized to random small values between -0.05 and 0.05 with each NN of the initial population using a different seed value for the random number generator. MSE on D_{train} is minimized for 100 epochs, whereupon the net with the best MSE on D_{val} is selected as training result. Convergence is very fast; in general, no more than 10 epochs are needed until training achieves an overall error smaller than 10% on D_{val}.

Optimization

The size of the initial population of NNs is set to 250. The size of the hidden layer is chosen to lie between 20 and 25 and is uniformly drawn for each NN of the initial population. After every training, magnitude-based pruning is applied to all NNs. One pruning step consists of the elimination of a percentage of weights; those 10% of weights are eliminated that have the smallest absolute value. All evaluations of optimization results are performed using the classification error (CE) on D_{test}. The result of the optimization is a statement about the network capacity needed for this particular problem class: it turns out that the number of connections can be reduced by approximately 55% while retaining optimal CEs, i.e. comparable to those obtained by repeatedly training the unoptimized

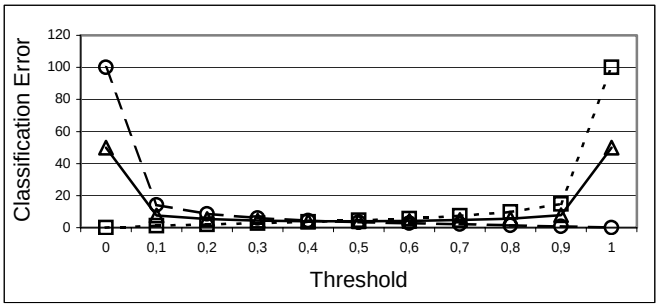


Figure 4.6: Error measures depending on applied decision threshold. False positives are depicted by circles, false negatives by squares. The overall classification error is indicated by triangles.

NNs with different initializations and choosing the best one as reference. At 10% of the original connections, NNs are still capable of an overall CE of about 95%. But, surprisingly, even NNs with fewer than 2% of the original connections are able to produce CEs of about 90%.

When talking about results, some terms should be clarified first since they are not used coherently in the literature: The false positive/negative rate is the percentage of positive/negative examples that are classified wrongly. Analogously, the true positive/negative rate denotes the percentage of correctly classified positive or negative examples. Fig. 4.5 (right) supports an assertion made earlier: the approximate independence of the CE on ROI size. What one can furthermore perceive is that CE actually falls below 3% when excluding ROIs smaller than 30 pixels. As shown by fig. 4.5 (left), such ROIs are quite common and their influence on the classification error is therefore notable, whereas they typically do not contain cars but artifacts produced by the initial detection. Fig. 4.6 shows the dependence of the classification error as well as the false positive and false negative rates as a function of the applied decision threshold. As can be perceived from the figures, the best overall classification error is 3,8%. Of course it can make sense to accept a higher overall error rate in order to minimize the false negative or false positive rates, depending on the demands of an application. Fig. 4.7 (left) shows representative optimization results. Notable is the clear trade-off between NN complexity and classification accuracy. Fig. 4.7 (right) shows the receiver-operator characteristic (ROC) of the best optimized NN.

4.8 Performance, benefits

On a Pentium II PC with 1 Ghz under Windows NT, using the MS Visual C 6.0 compiler, the largest NN from the optimization runs takes about 0.15 ms for one classification. The smallest optimized network that gives overall classification errors of under 6% takes 0.022 ms per classification. This results from the fact

that the net has only about 10% of the weights compared to the unoptimized NNs, with a corresponding order-of-magnitude increase in speed. The smallest NNs that yields an overall classification error of under 10% needs 10^{-4} ms for a classification. These extremely fast classification times make the optimized NNs applicable for brute-force search methods that scan the whole image at multiple scales to reliably detect all objects of interest (see chapter 8 for an elaboration of this idea). It should be noted, however, that it cannot be expected that this result is extensible to other objects than cars since the investigations suggests that the problem class is easily separable.

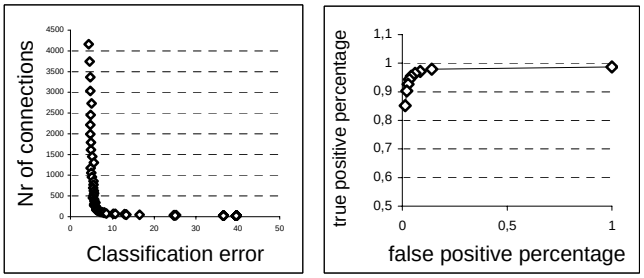


Figure 4.7: Left: Typical optimization results. Right: Receiver-operator characteristic of the best optimized NN

Together with the time needed for feature extraction, which is (averaged over all ROIs in the training sets) 0.3 ms, the object detection module takes between 0.3 ms and 0.45 ms for one operation, depending on the complexity of the used NN. Another point is that the optimization procedure generates a sequence of classifiers that can be applied successively to candidate ROIs, thus generating hypotheses of increasing reliability at negligible computational cost, since the feature set needs to be calculated only once. Classifiers of differing accuracy and complexity can be selected to enhance different modules of the system according to speed and accuracy requirements.

4.9 Conclusion

It was shown that car classification is a task that can be solved to very good accuracy and at high speed by NN classifiers. From the front of trade-off solutions that was obtained, NNs can be selected according to different demands on speed and classification accuracy. Extremal NNs containing only a few dozen connections are still able to give reasonable accuracy at negligible computational cost. These findings make it plain that car detection and classification can easily operate even under real-time constraints. In the face of the enormous variety of possible traffic situations, it is, however, not to be expected that the approach described here can lead to perfect detection accuracy under all circumstances. Therefore, further studies will be devoted to the issue of how the performance

of the whole driver assistance framework can be improved by the enhanced object detection, and how to make object detection more stable by considering additional information provided by other modules and information sources.

Chapter 5

Object detection with adaptive saliency maps

5.1 Summary

This chapter presents an initial detection architecture which is strongly inspired by the concepts of feature-based attention, biased competition and simultaneous bottom-up and top-down processing. It is not directly applicable as a real-time object detector since its computational complexity is rather high; nevertheless it is a powerful tool for building up more sophisticated mechanisms of object detection and scene understanding. The architecture is called an *adaptive saliency map* and is intended to perform bottom-up scene understanding and top-down guided target object search. It operates on gray-level images and integrates orientation, contrast and intensity features in a parametrized fashion over multiple scales. The parametrization can be adapted in order to put more emphasis on certain image features than on others; this mechanism aims to make target objects as salient as possible. This is achieved by competitive interactions strongly inspired by the "biased competition"-hypothesis of visual attention. The map output depends in a nonlinear and analytically intractable way on the parameters. Therefore, the evolutionary strategy of covariance matrix adaptation (CMA) is applied to optimize (learn) the parameters for a given target object class. Although analytical gradient information is not available (and, indeed, is not necessary for this method), convergence is quick, and significant increases in detection performance are observed when applying the saliency map to improve the initial car detection algorithm used in chapter 4.

5.2 Introduction

In this chapter, a new saliency map architecture for bottom-up scene analysis is presented. The aim is to quickly identify regions in an image which stand out from the background and merit a "closer look", possibly by an object recognition system. Conversely and more practically, parts of an image that are not interesting for object recognition are excluded from further analysis.

Since it would be impossible to do this for all types of conceivable objects, the model is parametrized, and (given sufficient training examples) there is a systematic way of learning parameter values which will enhance specific object classes while suppressing others, thus making target objects "pop out" from the background. In this way, a component of top-down modulation is introduced, enabling the model to attend to certain image features in a way similar to feature-based attention [16, 25] mechanisms in humans and primates. The implementation of the top-down process is strongly inspired by the "biased competition" [22, 74] account of visual attention. In human and primate vision there exists a symbiosis between top-down (concept-driven) and bottom-up (data-driven) processes. There is overwhelming evidence that almost all visual processing down to such low-level stages as the striate cortex can be strongly influenced by top-down signals, see e.g. [86] and references therein. In the absence of top-down modulation, tasks such as object categorization or the allocation of scan-path trajectories have been demonstrated to operate in a predominantly bottom-up fashion [100, 119].

5.2.1 Biological bottom-up scene analysis

This dichotomy has been especially thoroughly investigated experimentally for visual search experiments, in which (roughly speaking) subjects are required to find certain *target* items defined by particular features or combinations of features among others (so-called *distractors*). It turns out that certain targets are always found easily and quickly, whereas others require subjects to examine each item in a display separately, taking up more time in the process. It has been concluded that the knowledge of a target object's nature can influence low-level processing in such a way as to make desired objects stand out from a background composed of distractors. Nevertheless, there are limitations on this mechanism which mainly depend on the degree of dissimilarity between target and distractor items. The similarity measure, however, is highly complicated and reveals the detailed preprocessing that is performed upon a retinal image.

5.2.2 Computational modeling of bottom-up scene analysis

The model presented here is inspired by previous proposals for bottom-up scene analysis [66, 67, 135, 136]. It extends those models by a true "biased competition" mechanism, allowing for more efficient distractor suppression, and a new

learning algorithm suited to the highly nonlinear way in which the output now depends on the parameters.

The previous models put great emphasis on image processing and system-level simulation rather than the modeling of neural interactions themselves and have (among other purposes) been successfully applied to the rapid detection of vehicles in cluttered images [64] as well as to the explanation of scan-path trajectories of the human eye [102]. Common to all such models is the philosophy of integrating information from as many visual feature cues as possible. There also exist models which simulate the interaction between top-down and bottom-up interactions at an intermediate neural level [21] with intriguing results.

The presented model, however, is more in the spirit of [64, 67, 135, 136] because it places emphasis on correct image processing, integration of a wide range of visual feature cues and the nature of interactions existing between various visual cues.

Concerning the practical application of saliency maps, the most immediately apparent use is to restrict the search space of whole-image classifiers, i.e. classifiers that are applied at every location of an image over multiple scales in order to detect objects but bypass an initial detection stage. By using, e.g. convolutional network architectures (see [84] or chapter 6), this is well within the range of real-time processing on present-day standard computer hardware. Saliency maps can significantly speed up this process (as, for example, described in a recent publication [35]) as well as give additional hints as to whether a classification was a correct one.

5.2.3 Overview of the proposed model

First of all, terms will be precisely defined: input to the model comes from a video *image source* consisting of a sequence of *frames*. The model analyzes its input in a number of different ways termed *modalities*. For each modality, a number of *feature maps* is computed in a modality-dependent way (from single or multiple frames) at a predetermined number of spatial scales termed *pyramid levels*. Feature maps are then *downscaled* to a fixed size that is equal for all feature maps and *normalized* to a range between 0 and 1. This means determining the pixel of maximal response in all feature maps of each modality, and then dividing all maps of the modality by that value. For each feature map there exists a corresponding *weight*; a feature map is multiplied by its weight before being subjected to the *competition stage*, where it is combined with other feature maps of the same modality (possibly being at a different pyramid level, though) using center-surround differences. Thus, a set of new feature maps, termed *map responses* is obtained for each modality. Each pyramid level has its own predetermined *intrinsic scale*; each response is repeatedly filtered a certain number of times by a difference-of-Gaussians filter whose bandpass properties (see chapter 1) are chosen to enhance elements of the proper intrinsic size, thereby effectively implementing a rivalry between long-range lateral inhibition and short-range lateral excitation. As a last step, responses of all modalities are summed up and normalized to a range between 0 and 1, creating the *saliency*

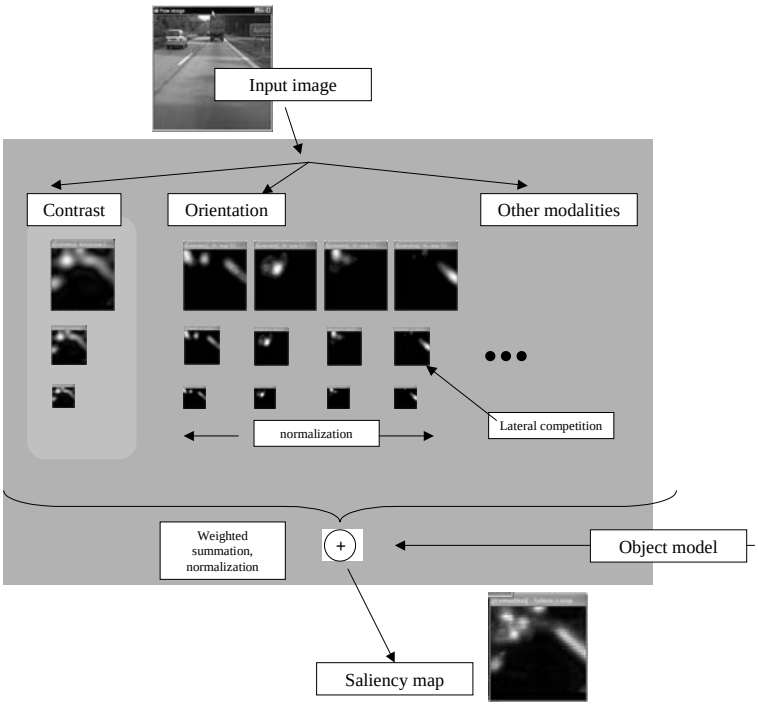


Figure 5.1: Overview over the saliency map model.

map computed from one frame. An overview over this process is given in fig. 5.1.

By ensuring that the feature maps are weighted *before* competition, a bias is introduced into the model: those feature maps with higher weights will have a better chance of winning local competitions. This is intended to be used for efficient distractor suppression: if a feature map wins the competition against another in a local neighborhood, usually the losing map will not appear at all in the final response but be strongly suppressed. Moreover, the competition is only local: other results may emerge in other regions of a response depending on the image source, but competition will be globally biased towards the feature map with higher weight.

The question arises immediately how to adapt the weights of the model in order to enhance certain object classes: For the time being, investigations were concerned with supervised learning strategies only. It is obvious that this architecture is unsuited for gradient-based learning, because the repeated center-surround filtering and subsequent normalization make it impossible to compute an analytical expression for error gradients. An important point of this chapter is therefore the choice of an appropriate learning strategy. For this purpose, the evolutionary strategy of Covariance Matrix Adaptation (CMA) [47] is employed. An analytical gradient is not required for CMA, yet it will be shown that CMA is nevertheless quick to converge. It can therefore be used efficiently as a learning

algorithm in general saliency map models and related tasks in computer vision.

This section presents the saliency map model and a learning strategy for it along with demonstrations of the model's capabilities and limitations. It will be the scope of further investigations to test and optimize the model in detail.

5.2.4 Outline

This chapter is structured as follows: In section 5.3, a detailed description of the proposed model as well as the used image processing algorithms is given. Section 5.4 deals with supervised learning of the model's weights, its possibilities, limitations and practical upshots. Section 5.5 briefly describes the evolutionary learning algorithm that is employed, outlining its necessity as well as its domains of application before giving a sketch of its working principles. In section 5.6, simple test scenarios and performance evaluation measures are described, which are evaluated in section 5.8. A discussion of results as well as further extensions and improvements to the model is conducted in section 5.9. Details on image processing methods can be found in section 5.10.

5.3 Saliency map architecture

As the basic architecture of the proposed model has already been described in section 5.2.3, the focus of this section will be on a detailed description of image processing techniques used for obtaining feature maps in each modality as well as a description of the competitive interactions between them. First of all, the common properties of all modalities will be described.

5.3.1 Modality-independent processing

Let us assume a total of M modalities. Each modality performs identical transformations of the image source at s_{max} spatial scales. At each scale, n feature maps are generated by applying several modality-dependent transformations (see section 5.3.2) to the image source. The results (whose dimensions will depend on the current spatial scale) are then downscaled to a fixed size (see chapter 1 for details on downscaling) which is identical for all feature maps[1]. Thus, a total of Mns_{max} feature maps will be generated. For each modality, all its feature maps will then be linearly normalized to a range between 0 and 1. For each spatial scale $s \in [0, \ldots, s_{max}]$, an intrinsic size i_S is defined; it is this size which determines the scale of local competition.

This is implemented in a twofold way as difference-of-Gaussian (doG or center-surround) filtering: first, in the process of generating map responses from feature maps, each feature map at level s is combined by a center-surround operation with all other feature maps of level s and, provided that it exists, level $s + 1$. Then, several local competition steps are applied the to resulting map responses. For generating those responses, all subsets of two feature maps

[1]The smoothing filter for downscaling is given in section 5.10

from the same modality are selected. One feature map is convolved with a Gaussian smoothing filter (its width determined from i_s, see section 5.10) and the other by a Gaussian smoothing filter with doubled standard deviation. Each convolution result is multiplied with the weight corresponding to the feature map it was computed from. Then, the pixel-wise difference between both results is computed and stored as a map response.

Map responses are subjected to center-surround filtering for f times, each result being input to the next center-surround operation. Pixel values which are smaller than zero are set to zero. The widths of the Gaussians are again determined by i_s according to section 5.10.

The rationale behind choosing this particular kind of competition is explained in section 5.3.3. Note that—since the widths of the center and the surround filters are large—downscaling can be performed first, and applying repeated center-surround filtering to the downscaled feature maps afterwards. This is done in order to increase the speed of the operation compared to filtering full-sized images.

Finally, all responses are summed up and normalized to a range between 0 and 1, producing the saliency map. For details, see section 5.10.

5.3.2 Modality-dependent processing

Inspired by the huge variety of feature-sensitive cells that have been found in the visual cortex of mammals [125], the proposed model aims to integrate as many local visual properties of the image source as possible when constructing the saliency map. Modalities can be added easily since no assumptions are made about them except that feature maps are required to have certain dimensions and be normalized to values between 0 and 1. For the time being, only three are implemented in order to demonstrate the capabilities of the model while keeping execution speed at a reasonable level. These modalities analyze the image locally for gradient energy (see chapter 1 or [10]), intensity and local orientation implemented by a steerable pyramid [32]. Among modalities that should definitely be included in future versions are color and primitive motion cues.

Gradient energy

This modality computes local contrast based on the nonlinear gradient energy operator (see 1.5.1) using the algorithm from [10]. One feature map $G^s(x, y)$ per level s is generated from the outputs of one-dimensional horizontal and vertical linear gradient filters. A threshold is imposed: pixels $G^s(x, y)$ with energies lower than the threshold are set to zero. One feature map is computed for each level. For details of this modality's processing see section 5.10.

Intensity

This modality simply returns a histogram-equalized gray-level version of the image source. Since in this particular implementation the image source already *is* a gray-level image, this modality computes little more than the histogram-equalized version of the image source. This means that one feature map is computed per level.

Local orientation

This modality computes local orientation and local spatial frequency from the outputs of steerable filters (see 5.10) of the order 2, which means that three feature maps are produced per level. It has been argued that preattentive human vision can only coarsely estimate local orientations in terms of a few orientations [134], and it has been attempted to model this by choosing three filters which respond best to horizontal, diagonal and vertical orientations. Although responding best to their preferred orientations, the used filters have coarse angular resolution so that any given orientation in the image will produce a signal in at least one filter, making sure that no orientations are disregarded. In effect, what each response measures is how much orientation and spatial frequency change in a local neighborhood.

5.3.3 Competition

Competition is implemented by repeated center-surround filtering of a map response. This is roughly analogous to neural mechanisms of short-range excitatory and long-range inhibitory connections and performs bandpass filtering. The number of repetitions is set to $f = 8$, and pixels which have values smaller than zero are set to zero after each step.

Motivation for this particular kind of implementation is the idea of creating competition between feature maps. This competition can be influenced by enhancing the weights of certain feature maps, giving them an initial advantage over others. Due to the iterative nature of the competition process, local responses which "lose" are eliminated from the final map response almost completely while the "winners" are enhanced correspondingly. This mechanism leads to a more efficient, almost complete suppression of local features associated with distractors and incorporates the principle of "biased competition" which is observed abundantly within the human and primate visual cortex [74]. Due to the parametrized nature of the model, this is no loss of generality since, after a change of weights, only the competition steps need to be repeated to obtain a different saliency map adapted for different targets and distractors.

5.4 Learning

The term "learning" refers to the task of adapting the weights of the proposed model in such a way as to enhance target objects while suppressing non-targets.

It should be clear that this will only be possible if targets and non-targets are sufficiently dissimilar in at least one response. It is important to keep in mind that the saliency map is not intended to perform classification: it therefore has no concept of local geometric structure of objects. Any enhancement or suppression of object classes happens purely by giving certain feature maps competitive advantages over others, i.e., raising or lowering their respective weights. Thus, for example, cars might be distinguished from lane markings but a distinction between different types of cars is not possible and indeed not intended.

There have been some attempts to implement such adaptivity into saliency maps, most notably in [66]. There, a model is described in which responses are weighted *after* they are computed (i.e. after competition), and an additive learning rule is used to update weights iteratively. While the reported results are promising, this model can only suppress non-targets linearly (by lowering the weights of feature maps associated with non-targets), whereas a more thorough suppression is desirable. Moreover, since competition is not influenced by the weighting, local features that have been eliminated by competition cannot enter into the final saliency map at all. In contrast to this, the model proposed here can suppress distractors more thoroughly by influencing the competition mechanism. Furthermore, by changing the weights and performing another competition step, hitherto suppressed features may appear in the saliency map at the cost of others. It is only fair to state that the competition step is computationally quite expensive (although modern graphics hardware can dramatically speed up the necessary convolutions), so there might be good reasons for avoiding it.

In [66] the concern is raised that learning the weights in a saliency map will reduce the general applicability of the map: up to a point, this is true, yet the freedom to choose weight values includes the case of setting all weights to equal values. Thus models without bias can be reproduced. Furthermore, it can often be very useful to have a more specialized model exhibiting object class sensitivity. Lastly, the parametrization can be changed at will, enabling the user to switch between different weight configurations exhibiting sensitivity to different target object classes.

Another challenge is the fact that the model (through the repeated center-surround filtering) not only depends in a strongly nonlinear way on the weights, but that the analytical form of this dependence is unavailable. It is therefore impossible to obtain the gradient of the saliency map performance (defined in section 5.4.2) with respect to the weights. Recently, evolutionary strategies have become increasingly popular for solving difficult optimization problems where no gradient information is available, such as the structure optimization of neural networks presented in chapter 7 or in [61]. An appropriate evolutionary learning strategy is therefore employed to overcome the difficulty of unavailable gradient information.

5.4.1 Training data

In order to make supervised learning methods applicable, training examples must be selected and labeled, i.e., assigned a number which encodes the class membership of every example. Since the goal of learning in saliency maps is to distinguish targets from non-targets, just two classes need to be included in the training data: "target" and "non-target". These classes are encoded by 0 for non-targets and 1 for targets. Two datasets are created, a *training set* D_{train} and a *validation set* D_{val}. Both contain an equal number N_{ex} of target and non-target examples. Whereas the training set is for learning, the validation set serves to verify the results using data which were not considered by the learning algorithm. In this way, mathematical statements about the generalization of a learning algorithm can be derived, see e.g. [82]. No such analysis was carried out for this investigation, and the validation set is simply used to select a weight vector (from all vectors produced in the course of training) that produces the best fitness on the validation set. Thus, a set of weights with good generalization behavior is selected. Each example $i \in \{1, \ldots, 2N_{ex}\}$ in a dataset contains a reference to an image img_i, a class label $l_i \in \{0, 1\}$ and the coordinates defining a rectangular region within that image, jointly denoted by r_i. The rectangular region encloses an object which belongs to the class encoded by l_i. For more details on the training databases that were used in simulations see section 5.6.

5.4.2 Fitness function

Central to the learning task is a quality measure (sometimes called *fitness function*) assigning a scalar *fitness value* to a given set of weights. Formalizing this, given a set of weights (more precisely, a number $x \in \mathbb{R}^w$ assuming a total number of w weights), the fitness function is defined as a mapping $f : \mathbb{R}^w \supset [0, 1]^w \mapsto \mathbb{R}^+$. An usual convention is to construct the fitness function in a way that assigns lower values to better (with respect to the learning task) weight vectors. Therefore the task of the learning algorithm is to find weight vectors that have the smallest possible fitness value.

The fitness function used here is a very simple one, based on the premise that targets should be enhanced and non-targets should be suppressed. Given a dataset D of training examples as described in section 5.4.1, the fitness is calculated by

$$f(D) = \sum_{i=1}^{N_{ex}} \sum_{\boldsymbol{x} \in\ r_i} |l_i - m_{\text{img}_i}(\boldsymbol{x})| \tag{5.1}$$

using the saliency map $m_{\text{img}_i}(\boldsymbol{x})$ calculated from the image img_i associated with an example.

5.5 Covariance matrix adaptation

Evolution strategies [5] are one of the main branches of evolutionary algorithms, i.e., a class of iterative, direct, randomized optimization methods inspired by

the principles of Darwinian evolution theory.

Due to space limitations, only the principles and performance considerations of the highly efficient evolutionary strategy of Covariance Matrix Adaptation are sketched here. Details can be found in [47].

CMA is an iterative, direct (only model responses, not gradients are needed) strategy repeatedly performing the following steps (termed *generations*): starting with a *parent population* consisting of μ *individuals*, each of which corresponds to a particular choice of real-valued parameters (in this case: map weights), an *offspring population* is generated in a partly probabilistic way. The offspring population contains $\lambda > \mu$ individuals. This process is termed *variation*. The fitness of the offspring is calculated and used for selecting μ individuals for the new parent population. In addition, the result of the fitness evaluations is used to adapt the distribution which governs variation, essentially a multidimensional Gaussian distribution. Using the new parent population, these steps are repeated until a termination criterion is met. In practice, the process continues until a predetermined number of iterations has been performed.

The adaptation of the distribution which governs variation is a key ingredient of CMA. The main idea is to alter the distribution in a deterministic way, increasing the probability that steps in parameter space that led to the current population are repeated. In doing this, the search path of the population over the past generations is taken into account, where the influence of previous time steps decreases exponentially.

The performance of CMA depends almost exclusively on the fitness evaluations of parent and offspring populations at each iteration: λ fitness evaluations are necessary for this purpose. Therefore the algorithm is efficient if the fitness evaluation can be performed efficiently. This is not the case in the present implementation; as a consequence, the number of examples that can be used for training the model is limited.

A solution to this lies in massive parallel processing: in principle, each fitness evaluation within one generation could be performed on a separate computer in a network since no interdependencies between single evaluations exist. The maximal gain in processing time is bounded by the number of offsprings in each generation. For the experiments described here, parameters of $\mu = 3$ and $\lambda = 21$ were used. This suggests that a 20-fold increase in performance can be achieved.

5.6 Test scenarios

For testing the performance of the proposed model and its benefits compared to other approaches, a test scenario based on a collection of real-world video sequences showing typical highway traffic scenes was created. All videos are gray-valued and have a resolution of 360x288 pixels. Experiments are (for the time being) limited to comparatively small numbers of examples for reasons given in 5.5. Lack of sufficient numbers of examples is of course a drawback since the statistical guarantees for generalization performance depend on this number. However, the purpose of the conducted experiments is rather to illu-

minate the working mechanisms and capabilities of the proposed model than to perform rigorous benchmarking. Furthermore, the experiments are intended to demonstrate how the detection of target objects can be significantly improved by learning specialized sets of weights, and this can be shown even with comparatively few examples.

A collection of four video sequences of highway traffic scenes was used for collecting training examples. Each sequence contains about 200 images. Target objects are cars; non-target objects consist of a selection of all conceivable other objects. A selection was chosen for non-targets which includes objects that are most similar to cars (e.g., lane borders or traffic signs) but also blank road parts and uniform textures like parts of trees. It is clear that this was done on an ad-hoc basis, but with the given restriction on the number of training examples, no other strategy was possible. By evaluating the performance of the learned sets of weights on an independent validation set, an effort is made to prevent overfitting and to show that the results are not restricted to the particular choice of image source. N_{ex} is set to 50, which is a value that allows reasonably fast learning. An experiment was conducted, again comparing fitness values with and without learning to each other. The visual inspection of the corresponding saliency maps is most instructive, please see fig. 5.2.

5.7 Performance evaluation

The evaluation of saliency map performance is performed using the initial car detection system described in chapter 4. The saliency map performance is measured by how much it can improve the detection results, i.e., how many false detections can be rejected and how many misses can be found. The reason for this rather roundabout way of performance testing is that a saliency map is not a classifier; it is intended to mark "interesting" regions based on the conjunction of simple locally dominant image features. As a consequence, the detection of regions containing non-target objects that have locally the same set of dominant features (for example, cars and rectangular traffic signs could be mentioned here) is nothing unusual; if one were to simply scan the saliency map for rectangular regions of certain sizes whose fitness value 5.1 exceeds a threshold, one would end up with a large number of false detections. By testing the saliency map on an already restricted (by the initial detection) set of regions, this problem is circumvented and, what is more, the practical use of a saliency map can be demonstrated by improving an already working system.

5.7.1 Performance measures

Two datasets of ROIs (regions of interest) are provided with each video sequence in the test set described in section 5.6: a *ground-truth dataset* and a dataset generated by the initial detection module described in chapter 4. Each dataset contains the coordinates of rectangular ROIs in single frames of one video sequence. The ROIs in the ground-truth dataset are manually generated

and correspond to image regions containing car back-views. It is important to keep in mind that the ground-truth dataset contains *all* car views present in each frame; by inference, any ROI within a frame that does not coincide with an ROI in the ground-truth dataset cannot contain a view of a car. By using the ground-truth dataset, it is therefore possible to group ROIs of one frame into several classes:

- *false positives* - ROIs produced by the initial detection which are not contained in the ground-truth dataset
- *true positives* - ROIs produced by the initial detection which are contained in the ground-truth dataset
- *misses* - ROIs in the ground-truth dataset but not found by the initial detection

The fitness value (5.1) which is based on an evaluation of the saliency map is calculated for each ROI of one frame; ROIs are categorized by checking whether the calculated ROI fitness exceeds a certain threshold. This categorization is now used to confirm or to correct the initial detection results for each type of ROI given in the list shown above. By varying the threshold that is applied, a several kinds of curves similar to receiver-operator-characteristics [126] can be obtained, from which the performance of the saliency map as well as the best threshold can be read off.

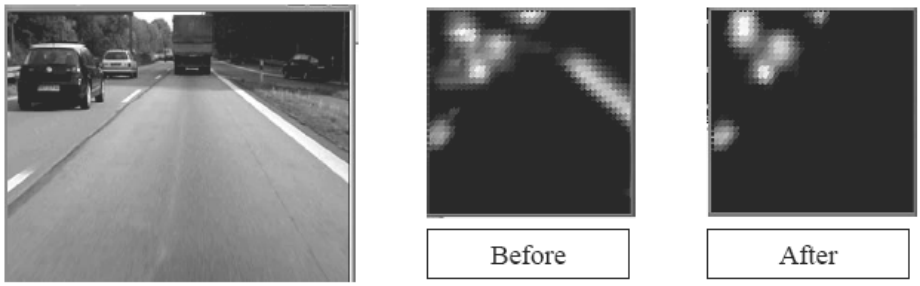


Figure 5.2: Distractor suppression by a trained saliency map. Left: original video image, middle: all saliency map weights set to 0.5, right: saliency map using learned weights. Observe that the right lane border is almost completely suppressed. This is possible because it is mainly encoded by the modality for local orientation; thus, a reduction of the weight for that particular orientation at the appropriate scale is sufficient for effective suppression.

5.8 Results

As one can perceive, it is possible to impose a saliency map threshold (e.g., 0.4) which strongly suppresses false positives while leaving true positives virtually

unchanged, as can be seen in the upper half of fig. 5.3. In addition, fig. 5.4 shows that at this threshold value, nearly all missed detections are "classified" as cars by the saliency map. The later fact cannot be easily exploited since in an online situation, the misses are unknown because ground-truth data is not available; nevertheless, is has been demonstrated that a re-examination of detected ROIs by a saliency map (with a suitable threshold) can significantly improve the accuracy of initial detection in an online scenario. For a visual-

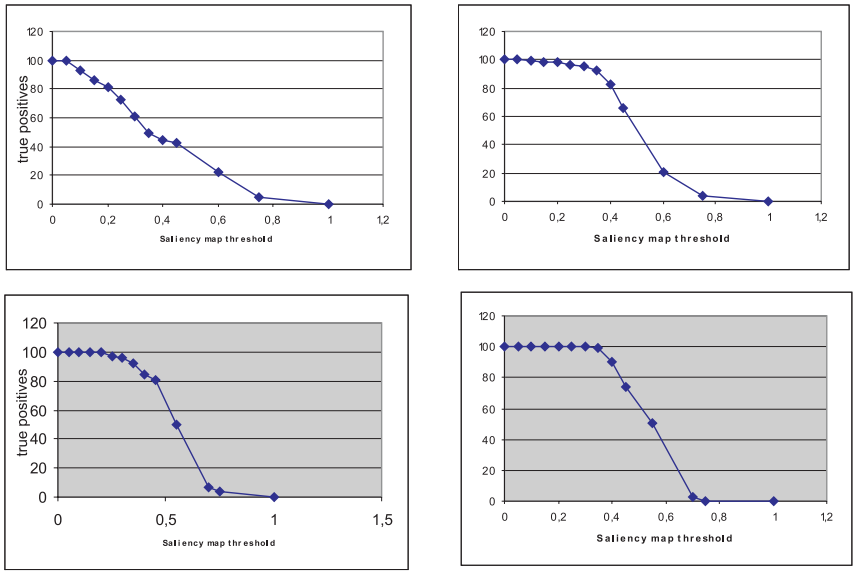


Figure 5.3: Effects of learning on detection performance. Upper half: Saliency map weights learned by CMA. Lower half: no learning, all weights have been set to 0.5. When the saliency map threshold is set to 0.4, the number of false positives can be reduced to 40% of its original value, whereas the number of true positives stays above 95% of its original value.

ization of the effects of learning, please refer to fig. 5.2, where one can clearly observe the suppression of stimuli which do not occur in target objects. On a 1.5 GHz Pentium II processor, about 5 frames per second could be processed when using the saliency map. This is not sufficient for real-time applications on present-day computers; it must be borne in mind, however, that the really time-consuming part of the saliency map calculation is the repeated center-surround filtering, which basically consists of repeated convolutions with large Gaussian filter masks. This is a task than can easily be performed by specialized hardware if it should be necessary. However, the presented saliency map model is not primarily intended for real-time applications, as shall be discussed in the next section.

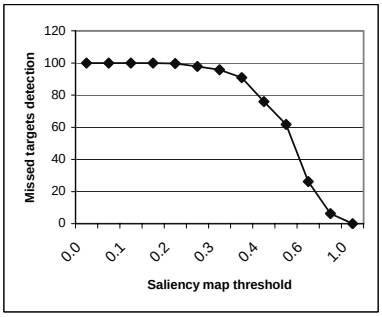


Figure 5.4: Saliency of missed detections. Note that virtually all missed ROIs are salient enough to be detected up to a threshold of 0.35.

5.9 Discussion

It has been demonstrated that it is possible to improve the results of initial detection by a suitable trained and parametrized saliency map as was described in this chapter. This, however, is just a first step, for the presented saliency map is a very general model of selective visual attention and should be used to develop more powerful and biologically motivated object detection algorithms (in contrast to those presented in chapter 3) rather than real-time applications. In order to achieve this, several extensions to the model may be considered: first of all, it should be investigated if appropriate object models (i.e., weight configurations) can be learned online. Apart from being more biologically realistic, it would also considerably simplify both implementation and application of the saliency map model. Another point worth considering is the question whether a component of space-based attention as found in psychophysical experiments [134] might not be employed with beneficial effects. Essentially, the saliency map model would then be able to enhance specific image *locations* instead of local features. This could be implemented easily by coarsely subdividing the image into quadratic non-overlapping parts, and giving each rectangular region its own set of weights. Such a model would learn to enhance or suppress certain features depending on their location within the image. The model is also suited to generate analogies to scan-path trajectories: after convergence, the strongest local activation within the map can be suppressed, and a subsequent center-surround step will produce a new locally strongest activation. If this procedure is repeated several times, the sequence of locations corresponding to locally strongest activations can be used for further recognition mechanisms; either, a classifier is applied at each point, or the geometrical structure of the sequence itself might be exploited (see [108] for more details about this concept). Further research opportunities using the presented saliency map model are suggested in chapter 8.

5.10 Technical details

The intrinsic sizes of the map are set in a way that allows the detection of objects (in this publication: cars) at all sizes in which they occur. Therefore, $i_s = \{25, 50, 100\}$ was chosen, which reflects the fact that cars come in sizes roughly between 20 and 100 pixels in diameter.

For efficiency reasons, only separable convolution filters (see chapter 1) are used throughout the implementation of the proposed model. Whenever a convolution is mentioned, it should be understood that this refers to *two* convolutions with each of the two one-dimensional filters which correspond to a two-dimensional separable filter mask.

5.10.1 Downsampling filters

Filtered images at all pyramid levels (ranging from 360x288 pixels to 90x72 pixels) which are produced in each modality are repeatedly downsampled by a factor of 2 to a fixed size of 45x36 pixels using a binomial smoothing filter (see chapter 1) of order 4 before each downsampling step. The filter mask is given by $\frac{1}{16}(1\ 4\ 6\ 4\ 1)$.

5.10.2 Center-surround filters

Basic components of difference-of-Gaussian (doG) filtering are two Gaussian filters at pyramid level s: an "on"-filter with standard deviation σ^s_{on} and an "off"-filter with standard deviation $\sigma^s_{off} = 2\sigma^s_{on}$. σ^s_{on} is chosen to be equal to half of the level's intrinsic size i_s (divided by the same factor that was used for downscaling) because i_s measures a diameter whereas σ^s_{on} measures a radius. A filter cut-off (see chapter 1) is applied at $2i_s$ pixels away from the zero point both for "on"- and "off"-filters. Zero-padding boundary conditions are applied whenever a filter exceeds a feature map's dimensions.

5.10.3 Steerable filters

Steerable filters of size 11 are used as described in [32]. Steerable filter are separable orientation-selective filters that (under certain conditions) approximate the orientation-selective Gabor filters described in chapter 1. The three basis functions G_{2a}, G_{2b}, G_{2c} are sampled at an interval of 1 pixel. The resulting filter masks are used as convolution filters with mirror boundary conditions whenever they exceed the source's dimensions. The three resulting filtered images at each pyramid level are then subjected to a pixel-wise conversion to absolute values, making strong negative responses count just as their positive counterparts.

5.10.4 Gradient filters

Two one-dimensional gradient filters of size 17 are applied in the x- and the y-direction using periodic boundary conditions (see chapter 1). From the results

at pyramid level s, the gradient energy $G^s(x, y)$ is computed as described in chapter 1. For efficiency reasons, it is chosen not to perform the square root operation of eqn. (1.14). Together with subsequent normalization, this amounts to a nonlinear amplification of high energy values.

Chapter 6

Object detection and feature base learning with sparse convolutional neural networks[1]

6.1 Summary

In this chapter, a different approach than in chaper 4 is pursued: the steps of feature extraction and initial detection can be performed by a new convolutional neural network model termed *sparse convolutional neural network* (SCNN). The model's suitability for real-time object detection in gray-valued, monocular video sequences is demonstrated. SCNNs are trained on "raw" gray values and are intended to perform feature selection as a part of training the neural network classifier. For this purpose, the learning rule is extended by an unsupervised component which performs a local nonlinear principal components analysis: in this way, meaningful and diverse properties can be computed from local image patches. The SCNN model can be used to train classifiers for different object classes which share a common first layer, i.e., a common preprocessing. This is of advantage since the preprocessing needs only to be calculated once for all classifiers. It is further demonstrated how SCNNs can be implemented by successive convolutions of the input image: scanning an image for objects at all possible locations is shown to be possible in real-time using this technique. Using this method, object detection can work without the initial detection step of chapter 4 while retaining real-time capability. It is shown that classification results are competitive to those presented in chapter 4.

[1]Some of the content of this chapter has been published in A. Gepperth. Visual object classification by sparse convolutional networks. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2006*. d-side publications, 2006. accepted.

6.2 Introduction

In many real-world classification tasks there is a need for classifiers that can learn from examples, such as neural networks (NNs) or support vector machines. Typically, the performance of such classifiers depends strongly on a suitable preprocessing of the input, but it is far from clear what characterizes an optimal preprocessing or if there even exists an optimal solution. Sometimes it is required that preprocessing should reduce the dimensionality of the input as far as possible, whereas another objective is to make preprocessing invariant to certain transformations of the input (typically translation, rotation and scaling are investigated in this context). The process of choosing an appropriate preprocessing transform is referred to as *feature selection*. In addition to constraints on error rates, the available processing time is usually bounded from above, too, especially in computer vision. Therefore, not only the accuracy of classifiers is important but also their execution speed.

Convolutional neural networks (CNNs) [83] were proposed to address all of these issues. They are specialized instances of multilayer perceptrons (MLPs) and thus essentially feed-forward NNs. Due to their connectivity, CNNs can be implemented by successive convolutions of an input image, permitting very high execution speed (see [37,118] for recent applications of CNNs). CNNs operating on unprocessed image data essentially *learn* a preprocessing transform, thus integrating feature selection into the training process.

In this chapter, a new convolutional neural network architecture termed *sparse convolutional neural network* (SCNN) is presented and its possibilities for object detection are explored. Since convolutional neural networks can be implemented using consecutive convolutions, whole-image search at multiple scales is possible in real-time on standard present-day computer hardware. Furthermore, the SCNN model is intended to perform feature selection from unprocessed image data: a hybrid supervised-unsupervised learning algorithm is described which computes meaningful and diverse features by the interplay of local nonlinear PCA and error minimization. Lastly, an algorithm for learning a common image representation that is shared by several SCNN object classifiers is described. The obvious advantage (especially when performing whole-image searches) of different classifiers using the same preprocessing is that preprocessing needs only to be performed once per image.

6.3 Classification problems

Most experiments described here are based on the problem of car classification in real-world video traffic scenes (see, e.g., [40]). For a few experiments, the problem of traffic sign classification is considered in addition. However, this problem is not a present focus of investigations, therefore training data are much less rigorously selected and tested, and results may not be very generalizable.

Object classifiers are trained to distinguish objects from background. Training data are generated manually by marking rectangular regions of interest

Figure 6.1: Positive and negative examples for the car/traffic sign object classes.

(ROIs) within a single video image that contain objects. The rectangles enclose an object as tightly as possible. Negative examples are also created manually, although that choice is very ambiguous. It was attempted to collect negative examples that are as similar to objects as possible. Some representative training examples for cars and traffic signs are shown in fig. 6.1. The requirement that the classification be invariant to certain transformations is encoded into the training examples. Let us define some notation: a *training example* consists of a *class label* and a *region of interest* (ROI) within a specified image. The ROI either does or does not enclose an object: to indicate this, the class label is set to 1 for an object and to -1 otherwise. A *training dataset* D contains N examples. Before using a dataset for training, a defined number of transformations is applied to the ROI of each example, creating the *transform dataset* D^{tr}.

First of all, the transformations to be applied must be specified as well as the degree of invariance which the classification should have with respect to these transformations. Let us assume that each transformation $f_t^{\alpha}, t \in [0, \ldots, T-1] : D \mapsto D^{\mathrm{tr}}$ can be continuously parametrized by a single parameter α, and that a total of T different transformations exists. Let $f_t^{\alpha=0}$ denote the identity transform. Then a limit $\alpha_t^{max} > 0$ must be specified, stating the range of parameters $I_t = [-\alpha_t^{max} \ldots \alpha_t^{max}]$ in which classification invariance should hold. Further assuming that all transforms commute (fulfilled for translation, rotation and scaling in two dimensions), we obtain a map

$$\tau : D \mapsto D^{\mathrm{tr}}; r \mapsto (f_1^{\alpha_1} \circ \cdots \circ f_T^{\alpha_T})(r), \alpha_k \in I_k, r \in D \ .$$

that is applied a defined number of times to each example in D. From the results, the transform dataset is created: it is therefore larger than the original dataset of examples. In the implementation presented here, a certain invariance to scaling and translation is required. Translation is modeled by two transformations, one for horizontal and one for vertical translations. The parameters α_x and α_y of both transformations are interpreted as the percentage of an ROI's width or height by which it should be shifted. The single scaling transform enlarges or reduces an ROI's width and height by a factor of α_{sc} while ensuring that the center of the ROI stays constant. In addition, it is required for the map τ that each transformation result must completely contain the original example.

D^{tr} is generated from labeled data by applying τ 9 times per example and uniformly drawing from the parameter intervals $I_{x,y}$, I_{sc} defined by $\alpha_{sc}^{max} = \sqrt{2}$,$\alpha_{x,y}^{max} = 10$. From the transform dataset, three disjunct datasets D_{train}, D_{val} and D_{test} are created which contain 2000 examples each, half of them positive. The image content within the ROIs is up- or downsampled to a fixed size of 25x25 pixels. Whenever necessary, appropriate smoothing and bicubic interpolation are performed. For later experiments, three additional car datasets

are created from D^{tr} where each ROI is shifted by 50% of its width to the left. The idea behind this classification task is to make detection more robust by checking if the "left-shifted" object classifier indeed finds half a car at the left of a detected car.

6.4 Sparse convolutional neural network classifiers

Like the original proposal [83] they are derived from, SCNNs are feed-forward neural networks with local receptive fields (see fig. 6.2). However, the connection structure in SCNNs has been considerably modified as compared to [83]. The proposed model is simpler and can —once trained— be tested using existing software for simulating multilayer perceptrons. Furthermore, the issue of obtaining meaningful and diverse features is addressed using a direct approach. The original CNN model attempts to achieve this by connecting hidden layers only to certain (not all) succeeding layers, which has been experimentally shown to lead to dissimilar feature maps. It is unknown, though, what effect the global network structure has on this mechanism and how many experimental trials are necessary for this mechanism to work. In the SCNN model (see fig. 6.2), feature complexity and diversity are enforced by additional unsupervised terms in the learning algorithm. They cause outputs of different feature maps at the same image location to be (nonlinearly) decorrelated and to have extremal variance in a way very similar to nonlinear principal components analysis [56]. Employed principles are gradient-based variance maximization of neuron outputs, decorrelation and weight vector normalization. The SCNN model has an input layer of fixed dimension, one or more hidden layers, and an output layer containing a single element. Each layer receives input from one other layer (the preceding one) and projects to a single layer (the succeeding one, see also fig. 6.2).

6.4.1 Network model

Since SCNNs are specialized instances of multilayer perceptrons, the network structure is discussed without reference to the implementation as successive convolutions. Only at the end of this section, some constraints arising from this implementation are discussed.

Connectivity

A layer l having dimensions $L_l^x \times L_l^y$ is composed of identical *cells* of neurons of dimension $C_l^x \times C_l^y$. Thus, a neuron can be assigned coordinates $\boldsymbol{n} = (l, \boldsymbol{c}, \boldsymbol{i})$, where $\boldsymbol{c}$ denotes the two-dimensional index of the cell within layer l, and $\boldsymbol{i}$ the neuron's coordinate within its cell. Within one cell, each neuron is connected to the same rectangular patch of neurons in layer $l-1$ which is termed a neuron's *receptive field* (RF). Receptive fields in layer $l-1$ can overlap in x- and y-direction by $O_{l-1}^x \times O_{l-1}^y$. The set of all weights connecting a neuron to its RF

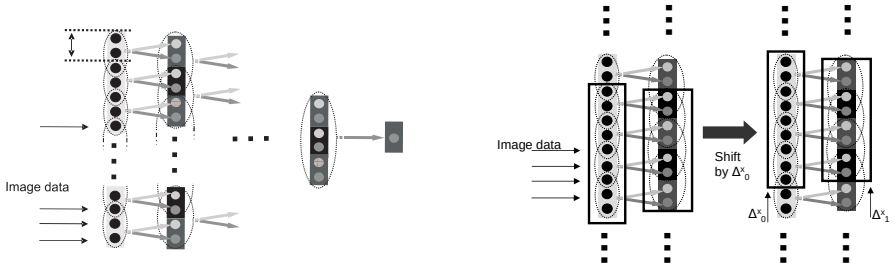


Figure 6.2: Left: Sketch of the SCNN network model (cross-section, y-dimension is not shown). Receptive fields are drawn in by dotted ellipses, cells are indicated by the alternation of darker and brighter backgrounds. Input filters connecting neurons to their receptive fields are shown as arrows in different shades of gray which match the shade of the destination neuron they project to in the next layer. Arrows of the same shade of gray represent *equivalent* input filters, see text for details. In addition, the *step size* Δ_0^x is shown: it is the number of input neurons (i.e., pixels) by which the classifier samples the input image in the x-direction when performing whole-image searches. Note the local connectivity of SCNNs: each neuron receives input only from a part of the neurons of the preceding layer. Right: Details on whole-image search with SCNNs (y dimension is not shown). The diagram shows how the classifier (represented by rectangles) can be applied at shifted positions of a whole image. When the classifier is shifted in the input layer by Δ_0^x, it simply moves on by one cell in the layer 1 (and all subsequent layers), too. From this, it is clear that the corresponding step size Δ_1^x must be a multiple of layer 1 cell sizes, or the shiftability property is lost. Equivalent conditions must be fulfilled in all subsequent layers, too, by the same reasoning. Assuming that the whole image has been processed by successive filtering and recombination, the classifier can be shifted in steps of Δ_0^x over the whole image without the need for additional calculations.

is denoted *input filter*. Since it is in one-to-one correspondence to a RF, it can naturally be arranged in a rectangular scheme with dimensionality $I_{l-1}^x \times I_{l-1}^y$ which is identical to that of the RF. Connection strengths are denoted by $w_{\boldsymbol{n}\boldsymbol{n}'}$ where $\boldsymbol{n}$ specifies the coordinates of the destination neuron and $\boldsymbol{n}'$ those of the source neuron. Please refer to fig. 6.2 for a visualization. Each neuron (except for those in the input layer) is connected by a trainable weight to a bias neuron whose activation is constant (here: 1.0).

Constraints

The first set of constraints comes from the geometrical consistency of the SCNN. Trivially, given a layer l, L_l^x, L_l^y must be integer multiples of C_l^x, C_l^y. Furthermore, the number of input filters in layer $l-1$ must be identical to the number

of cells in layer l. Thus, we get two conditions

$$L_l^{x,y} = kC_l^{x,y},\ k \in \mathbb{N}^+ \tag{6.1}$$

$$\frac{L_l^{x,y}}{C_l^{x,y}} = \frac{L_{l-1}^{x,y} - I_{l-1}^{x,y}}{I_{l-1}^{x,y} - O_{l-1}^{x,y}}\ . \tag{6.2}$$

A weight-sharing constraint enters via the requirement that neurons within a layer l, having the same within-cell coordinates $\boldsymbol{i}$ but being connected to different RFs, must have identical input filters. It is this constraint which allows to implement a network run by a series of convolutions. In contrast, each neuron in one cell is allowed to be connected to the common RF by different filters than the other neurons in that cell. Effectively, the size of one cell, $C_l^x \times C_l^y$, specifies the number of convolution filters necessary for the simulation of each layer, whereas the size of receptive fields (equal to input filter size $I_{l-1}^x \times I_{l-1}^y$) determines the dimensions of the convolution filters. For each layer l, sets of weights that are required to have the same value by the weight-sharing property are called *equivalent*. Obviously it is desirable to obtain a trained SCNN which requires as few convolution filters as possible while maintaining high classification accuracy.

A further constraint comes from the implementation that is used for whole-image search (see section 6.5) although it is not necessary for the simulation of the SCNN model per se: it requires that step sizes Δ_l^x, Δ_l^y in layer l (i.e., the differences between the size of input filters projecting to layer $l+1$ and their overlap) must be integer multiples of that layer's cell sizes. Fig. 6.2 illustrates this particular constraint. In precise terms:

$$\Delta_l^{x,y} \equiv I_l^{x,y} - O_l^{x,y} = kC_l^{x,y},\ k \in \mathbb{N}\ . \tag{6.3}$$

Activation functions

The activity $A_{\boldsymbol{n}}$ of a neuron is calculated from the activities of its RF and the weight values in its input filter as $A_{\boldsymbol{n}} = \sigma(\sum_{\boldsymbol{n}' \in \mathrm{RF}} A_{\boldsymbol{n}'} w_{\boldsymbol{n}\boldsymbol{n}'})$ using the sigmoidal activation function $\sigma(x) = \frac{x}{1+|x|}$.

6.4.2 Learning in SCNNs

Initially, all weights are initialized to small random values between -0.01 and 0.01 (see [103] for a motivation of this initialization). Then, a weight-sharing step is performed: for each layer l, the average of each set of equivalent weights is computed. Subsequently, all equivalent weights within layer l are set to their previously computed average value. In this way, all equivalent weights have identical values at the start of training. During each learning step or *epoch*, all weights of the SCNN are treated as if they were independent. An improved variant of the well-known Rprop learning algorithm (IRprop+, see [58]) is applied to the SCNN using dataset D_{train} for 80 epochs. After each epoch, the weight-sharing condition is enforced as described before. Note that weight-sharing is enforced separately for the bias weights of each layer.

Figure 6.3: Input filters in the input layer of a trained SCNN. The images show input filters of identical SCNNs trained with different learning rules: MSE gradient (left) and hybrid learning rule described in the text (right). Note that many filters in the left images are almost identical whereas in the right images, such redundancy does not occur.

The mean squared error (MSE) is calculated as $E_{\mathrm{MSE}}(D) = \frac{1}{|D|}\sum_{p=0}^{|D|}(A_p^{out} - c_p)^2$ using a dataset D. It uses the class label c_p of pattern p and the activation A_p^{out} of the CNN's output neuron in response to pattern p. The learning rule for each weight is composed of the usual MSE-minimizing term plus an additional unsupervised term. The additional term is an approximation of Oja's nonlinear subspace rule [56]:

$$w_{\boldsymbol{n'n}}^{\mathrm{new}} = w_{\boldsymbol{n'n}} + \gamma A_{\boldsymbol{n}}(A_{\boldsymbol{n'}} - \sum_{\boldsymbol{j}\in\mathrm{cell}(\boldsymbol{n})} A_{\boldsymbol{j}} w_{\boldsymbol{n'j}}) \tag{6.4}$$

where γ is a small positive constant and the sum on the right-hand side of the equation runs over all neurons in the same cell as $\boldsymbol{n}$. Please see chapter 2 for goals and background of this and other PCA learning rules.[2]

During training, model selection is performed using $E_{\mathrm{MSE}}(D_{\mathrm{val}})$ alone. When evaluating the performance of a trained network, the classification error $\mathrm{CE}(D_{\mathrm{test}})$ is used. It is defined as $\mathrm{CE}(D) = 1 - \frac{1}{|D|}\sum_{p=0}^{|D|}\theta(A_p^{out} - \tau)$, where θ denotes the step function and τ a threshold assigned to each NN (always taken to be 0).

A few comments on the chosen learning algorithm are in order: a local nonlinear principal components analysis (please see chapter 2 for a comprehensive introduction) is performed within each receptive field, but modified by the MSE gradient. The unsupervised part of the learning rule selects those weight vectors which capture the largest possible part of their input's variance. It also leads to orthonormal input filters with an euclidean norm of 1.0. The learning algorithm is an extension of the algorithm given in [39] where only orthonormalization was performed (not by gradient descent but operating directly on the weights). Due to unsupervised learning, neurons within a cell capture a part of the variance of their input that is maximally large. Furthermore, the neurons' outputs are decorrelated which leads to the formation of dissimilar input filters (see fig. 6.3).
For weights connecting to the output neuron, the unsupervised term in the learning rule is not considered because it interferes too much with correct classification, i.e., the MSE-minimizing term.

[2] With the substitutions $A_{\boldsymbol{n}} \to \sigma(W\boldsymbol{x})$, $A_{\boldsymbol{n'}} \to \boldsymbol{x}$ and by writing out the matrix eqn. (2.11) in component notation, it is easy to see that eqns. (6.4) and (2.11) are equivalent.

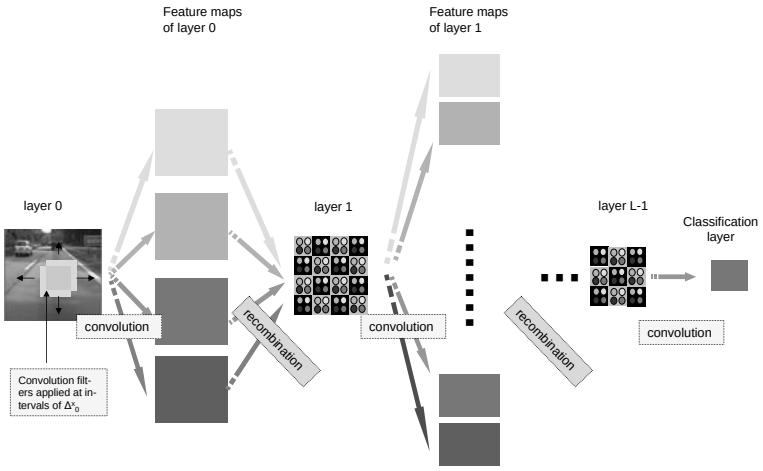


Figure 6.4: Sketch of the architecture to implement whole-image searches with the SCNN model. The input layer of the SCNN now consists of the whole image, and successive layers are correspondingly enlarged. Input filters of the SCNN translate into convolution filters: convolution results of a layer with its input filters are called *feature maps*. The recombination of feature maps into the next layer mirrors the connectivity of the SCNN model. Identical shades of gray of hidden layer neurons and feature maps indicate this. Instead of converging to a single output neuron, the SCNN now converges to a classification layer where each neuron (or pixel) represents the output of the SCNN classifier applied at certain image location. Note that symbols and shades of gray have been chosen in accordance with fig. 6.2 in order to convey the one-to-one relationship.

6.5 A convolutional architecture for whole-image search

The neural network architecture described in the previous sections is particularly suited, due to the weight-sharing constraint, for fast implementation by means of convolution filters (see, e.g., [69] for an introduction). However, it is possible to achieve far greater speed gains when considering *whole-image search*, i.e., the application of a fixed-size classifier at every conceivable position within an image, possibly at several scales. In this context, CNN architectures like the SCNN model have the tremendous advantage that convolutions for overlapping classifiers need only be computed once. This can be understood by considering that input filters in the SCNN do not depend on their spatial position within a layer due to weight-sharing. By inference, the whole image needs to be convolved *only once* with all input filters in order to produce a classification result at each position within an image. Please see fig. 6.4 for details of the convolutional architecture and fig. 6.2 for details on whole-image search with this architecture. Furthermore, since the input image is usually subsampled by the input filters of

the first network layer (always the case when $\Delta_0^x \geq 2$, see fig. 6.2), only the convolutions with these input filters contribute significantly to the total processing time. The whole-image classification problem then reduces to filtering with a limited number of (usually non-separable) filters; if real-time performance is desired, the mask size should be small (typically, sizes of 5, 7 or 9 are chosen). As was already mentioned, the SCNN model presented here belongs to the class of convolutional neural networks which were originally proposed in [83]. A crucial difference is that no implicit subsampling of feature maps is performed, whereas in [83], feature maps are successively filtered and subsampled until they converge onto one neuron, the output of which is combined with similar neurons to form a classification output. In the SCNN model, subsampling is performed if the step sizes in one layer are chosen larger than 1, but the choice of subsampling filters is not defined a priori (i.e., Gaussian smoothing) but learned by the network, too. It should be stressed here that subsampling *is* possible in the SCNN model, but at this point it seems more practical to let the SCNN *learn* the downsampling filters as well. A second difference is that feature maps are recombined into layers in the SCNN model after each convolution, whereas in [83], feature map outputs are recombined only in the last layer of the network. Recombination after each convolution inflicts a small computational cost, but it can be expected that it results in more reliable detection of conjunction features similar to the object detection architectures of [105, 128]. Lastly, there exists a direct mechanism of enforcing diversity among the learned features of the SCNN, which guarantees without the need for additional experiments that informative and non-redundant features are learned which, in addition, capture a significant part of the input's local variance (by the local PCA property).

6.6 Feature base learning

An interesting application is motivated by the observation made in the previous section that the computational load is biggest during the simulation of the first layer. If two networks had identical processing in that layer, they could be used simultaneously for whole-image search while the convolutions of the input image would only need to be computed once. Stated in different terms, it would be interesting to find out if there is a *common feature base* for two or more object classes, i.e., a preprocessing of the input which is suited for representing all of the object classes under consideration. This is inspired by the fact that cortical area V1 in the visual cortex of mammals performs a limited number of very basic preprocessing steps (see, e.g. [28]). Therefore, it is investigated if and how a common feature base for N object classes can be learned only from available examples. It is tested by experiment whether it is possible to achieve classification rates comparable to those attained when training classifiers separately. In the formalism of SCNNs, there exists a straightforward approach: Basically, N networks are trained independently from each other using methods given in section 6.4.2, but after each iteration of the learning algorithm, a weight-sharing

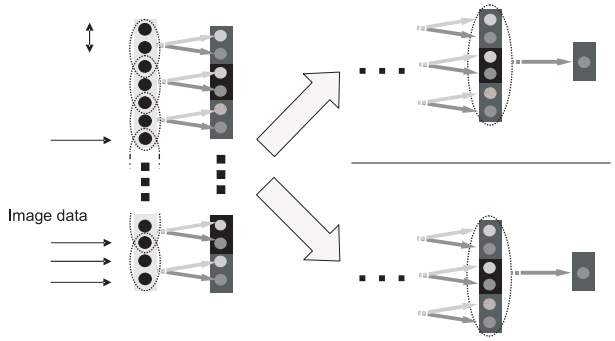


Figure 6.5: Exemplary SCNN architecture for feature base learning. The horizontal line indicates that the processing streams that converge onto the two (or more) output neurons do not have any connections in common.

constraint is enforced between the filters in the input layers of all networks.[3] An alternative interpretation is a single network which converges onto N output neurons by noninterfering processing streams from a common first hidden layer. Please see fig. 6.5 for a visualization. Model selection is performed using the sum $\sum_{i=0}^{N} E^i_{\mathrm{MSE}}(D^i_{\mathrm{val}})$ of the mean squared errors of each individual classifier on its validation dataset D^i_{val}.

6.7 Experiments

Two types of experiments are conducted: the speed of the system is tested using selected SCNN topologies, and assessments of topology-dependent classification performance are conducted. The latter task is performed off-line using training data mainly from the car classification task (see fig. 6.1).

6.7.1 Off-line classification performance of single SCNNs

Due to the weight-sharing constraint (see section 6.4.2), the number of free parameters in an SCNN is greatly reduced. The choice of an appropriate topology is therefore crucial since the number of free parameters in the network depends directly on it. Since the correct choice of NN topology for a given classification task is still, in general, an unsolved problem, a number of experiments was conducted to identify suitable topologies. The search space can be reduced by the requirement that small input filters should be used in the input layer, as well as by the architectural constraints (6.1), (6.2) and (6.3) which SCNNs must obey.

In each experiment, a certain SCNN topology is trained 6 times using a different random seed each time, and the best classification result $\mathrm{CE}(D_{\mathrm{test}})$ is taken to be a measure of that topology's learning capacity. As a baseline

[3] Note that the dimensions of the input layers need not be identical, only the dimensions of the input filters in the input layer.

Table 6.1: Best classification errors of various SCNN topologies for cars (C), cars shifted left (L) and traffic signs (TS) (errors are given in percent). Cells and layers are quadratic so only one dimension of their sizes is given (in columns "layers" and "filters"). In row 0, the result for two fully connected reference networks of MLP type is given. SCNNs 1-5 demonstrate the effects of varying input filter sizes and numbers. Notable is the improvement when allowing 4x4 input filters as in SCNN 5. The last two rows give the results for the feature base learning (using topology 4, see text) of two and three object classes.

Nr.	layers	filters	nr.filters	conn.	free param.	$min_{\text{exp}}\text{CE}(D_{\text{test}})$
0	25-5-1	(25)	-	15650	15650	5.5 (C), 5.4 (L), 6.7(TS)
1	25-9-1	21-9-x	3-1-x	36162	4050	6.8
2	25-18-1	9-18-x	2-1-x	26568	648	7.8
3	25-30-1	7-30-x	3-1-x	45000	1341	6.7
4	25-22-1	5-22-x	2-1-x	12584	584	6.8(C),5.8(L),11.4(TS)
5	25-44-1	5-44-x	4-1-x	50336	2336	6.3
6	feature base learning using SCNN 4					7.3(C),5.9 (L), 10.9 (TS)
7	feature base learning using SCNN 4					6.5(C),11.0 (TS)

for SCNN performance, a fully connected NN with one hidden layer is trained on the car classification problem in the same way. Table 6.1 gives an overview over representative SCNN topologies as well as the fully connected reference networks. When considering SCNNs with one hidden layer, two ways to improve classification results were identified: increasing the size, or alternatively the number of input filters in the input layer. Obviously, both operations lead to a larger number of free parameters. The best topology found in this way has filter sizes of 5x5 pixels in the input layer, yet it is not quite compatible with real-time requirements since it requires 16 convolutions of the input image in the input layer alone. It uses 80000 connections, although the actual number of free parameters is 2384[4]. It is notable that the classification performance is only slightly worse than that of the reference network despite the fact that the number of free parameters is much lower.[5] If real-time capability is desired, SCNN 4 is the topology to choose. Although using a much smaller number of connections and free parameters than topology 5, it achieves only slightly worse classification performance. Please see section 6.7.3 for speed measurements.

For unknown reasons, the inclusion of more hidden layers did not improve performance. Many-layered topologies were constructed by adding new layers onto well-performing SCNNs with one hidden layer. Notable was much slower overall learning convergence. It is therefore conceivable that training was not conducted sufficiently long. More research will have to be applied in order to shed light on this particular point.

[4]Note that it is the number of free parameters which determines the speed of whole-image classification

[5]It was also shown that SCNN performance is slightly superior to that of an MLP with identical connectivity as well as an SCNN using supervised learning only.

6.7.2 Feature base learning results

SCNN topology 4 given in table 6.1 is used for learning a common feature base for cars and traffic signs. It is not the best-performing topology that was found but comes very close to it; what is more, it allows real-time operation. Training is performed using the algorithm given in section 6.6. Results are given in table 6.1. It is evident that classification results are comparable to those of classifiers trained separately on their respective tasks. Observe that the feature base result for traffic signs has to be compared to traffic sign results of topology 4 in table 6.1, not to the reference network performance: the goal was to show that the performance of the individual classifiers can be reproduced by feature base learning. When choosing SCNN topologies with larger input filters, the performance of the reference network can be approached for traffic signs, too.

6.7.3 Online performance

All tests were conducted using a 1.86Mhz Centrino processor. Images had a size of 360x288 pixels; convolutions were implemented in C++, and no use was made of the capabilities of the graphics hardware. Classification was performed at three spatial scales for each frame, where each scaled image was obtained by smoothing with a size-5 binomial filter and downsampling by a factor of 2. The best-performing SCNN topology 5 given in table 6.1 allows a frame rate of 7 frames per second (fps), whereas SCNN 4 allows 22 fps at the price of slightly inferior classification performance. When using three classifiers of topology 4 sharing a common preprocessing, a speed of 19 fps is attainable.

6.7.4 Learned –vs– designed visual features

The SOE feature extraction algorithm presented in chapter 4 has been designed "by hand" to represent the object class of cars optimally. It is therefore interesting how the classification accuracies achieved by the SCNN model compare to the object classifier presented in chapter 4. It uses the SOE algorithm to compute a feature set which is then classified by a fully-connected MLP with one hidden layer of 20 neurons.

The classification error of approximately 3 % reported in chapter 4 is not directly comparable to the errors listed in table 6.1 because in chapter 4, no transformations (see section 6.3) are applied to the training examples prior to generating the training databases. This is omitted because an initial object detection system is assumed to be available which localizes cars with good accuracy before the classifier is applied. Furthermore, invariance to scaling and translation is encoded in the SOE algorithm. For both reasons, it is not necessary to take the requirement of invariance to translation and scaling into account when generating the training databases.

The classification error on D_{test} turns out to be 4.8% when training databases are generated according to section 6.3 (using the SOE method for feature extraction), and the classifier described above is trained using these databases.

This shows clearly that feature design does have its merits; on the other hand, one can argue that the results are obtained using a much higher number of free parameters (about 5000) than in the best SCNN of table 6.1, and that the difference is small.

Indeed, in chapter 7 it will be shown that the NNs from chapter 4 can be optimized to a size of about 2000 connections without losing classification accuracy. This result seems to suggest that the number of *necessary* connections in the NN is roughly equal to the number of free parameters of the best-performing SCNN. This is not the case, since the NN optimization is performed using the same training data as used in chapter 4. It can be safely conjectured that the number of necessary connections grows when performing classification on the much more difficult databases used in this chapter.

6.8 Discussion

The new convolutional network model is interesting in several respects: on the one hand, it demonstrates a successful combination of supervised and unsupervised learning rules; on the other hand, it offers very interesting possibilities for practical applications. Due to its real-time capability and the ability to search images simultaneously for several object classes using a trainable common preprocessing, it is suited for applications where scene analysis is performed, which consists usually of the recognition of more than one type of object. The driving idea behind the SCNN model was to reduce the need for "manual" feature design, and it could be shown that, although the results of manual feature design could not be surpassed, they were at least roughly equaled.

With SCNNs, *some* prior knowledge must still be provided in the form of the network topology: if it is known that, for example, that features of a certain size are characteristic of an object class, the input filters should be chosen accordingly. In many cases, input filter and step sizes are constrained by real-time requirements; once input filter and step sizes are fixed, the SCNN topology constraints are sufficient for removing most of the remaining ambiguities. As with all NNs, the correct choice of topology is an unsolved problem, although in practice one can simply take the SCNN with the largest number of parameters that is compatible with the constraints of the relevant application. As has been demonstrated, increasing the number of free parameters tends to improve classification performance (at least for SCNNs with one hidden layer).

The issue of extending SCNN topology successfully to more than one hidden layer is a current research topic; above all, it is important to know how topologies must be chosen such that filters in the hidden layers can be small (if speed is an issue). SCNNs with two or more hidden layers may be much more powerful in capturing local combination features; furthermore, it is intuitive that feature base learning can profit greatly from such topologies. The SCNN model itself could also be extended; in particular, shortcut connections which bypass one or more layers, and subsampling layers (as in LeCun's original proposal) suggest themselves here. From a theoretical point of view, a detailed examination of the

interplay between the supervised and unsupervised terms in the learning rule would be interesting; the relation of learned SCNN input filters to independent components seems to be worth investigating. Last but not least, it is intended to use SCNN classifiers (possibly in conjunction with other modules) to build robust and fast object detection systems that reliably work in practice.

Chapter 7

Structure optimization of object classifiers[1]

7.1 Summary

This chapter aims to optimize a different aspect of object detection, namely the connection structure of neural network classifiers which are used for the confirmation of initial object hypotheses. The issue of feature design is not touched here, and the structure of neural networks is optimized using a fixed preprocessing. For the purpose of finding suitable network structures, multi-objective evolutionary optimization of neural networks is applied to two real world problems, car and face classification. The possibly conflicting requirements on the NNs are speed and classification accuracy, both of which can enhance the embedding systems as a whole. The results are compared to the outcome of a greedy optimization heuristic (magnitude-based pruning) coupled with a multi-objective performance evaluation. For the car classification problem, magnitude-based pruning yields competitive results, whereas for the more difficult face classification, it is found that the evolutionary approach to NN design is clearly preferable

[1]Some of the content of this chapter has been published in: A. Gepperth and S. Roth. Applications of multi-objective structure optimization. *Neurocomputing*, (69):701–713, 2006. Design and implementation of the evolutionary multi-objective optimization algorithm and the implementation of multi-objective performance measures was done by S.Roth.

7.2 Introduction

The work presented here deals with the optimization of feed-forward neural network (NN) classifiers that support real world object detection tasks. The attribute "real world" is intended to emphasize that these are not toy problems but systems that are presently being used in commercial products, or will be in the near future. Here, error tolerances are usually quite restrictive. When considering existing applications of real world object detection systems it is obvious that imperfect object detection can lead to serious (possibly fatal, e.g., in automotive applications) problems, thus imposing a tight constraint on tolerable errors rates. To make matters even more difficult, detection should not only be near-perfect but also capable of real-time operation, placing strong bounds on the complexity of the methods that are used. It is intuitively clear that these constraints will not always coexist peacefully, and that methods must be developed to design and optimize systems in the presence of conflicting objectives.

Real world object detection

Many commercial object detection tasks for which solutions meeting the abovementioned constraints (low classification error in real-time operation) have been proposed fall into the domain of advanced driver assistance systems. These systems typically require the detection of pedestrians [99, 114, 122], vehicles [31,40,112], lane borders [24], and traffic signs [2] to ensure the "intelligent vehicles" can construct an adequately complete representation of their surroundings and (possibly) take the appropriate actions. Other domains, requiring successful face recognition in particular, can be found in commercial applications such as content-based image retrieval, video coding, video conferencing, automatic video surveillance of a crowd, intelligent human-computer interfaces, and identity authentication. Face detection is the inevitable first step in face recognition, aiming at localizing and extracting the regions of a video stream which contain a view of a human face. Overviews of examples, problems and approaches in the research domain of face detection can be found in [52, 139].

Thus, the problems of detecting cars and human faces are important examples of current research in visual object detection. Usually, architectures are broadly motivated by biological visual search methods [14, 137]: a fast initial detection stage localizes likely target object locations by examining easily computable visual features, whereas a more detailed analysis is then performed on all candidate regions or object hypotheses that have been formed in the initial detection stage. This chapter is concerned with the classification of hypotheses, that is, the decision whether a given object hypothesis actually corresponds to a relevant object type. Based on previous investigations (see chapter 4 and [132]), the problems of car and face detection are discussed in-depth.

Neural classifiers

The task of optimizing the weights and the structure of the NNs used for face and car classification in the ViisageFaceFINDER® system [120] and the car detection system described in [40] is addressed. In both cases, one goal is to increase the speed of the neural classifier, because faster classification allows for a more thorough scanning of the image, possibly leading to improved recognition. Another goal is of course to enhance the accuracy of the NN classifiers. It cannot be expected that these two requirements can be achieved independently and simultaneously.

Feed-forward neural networks have proven to be powerful tools in pattern recognition [141]. Especially in the domain of visual object detection the competitiveness of NNs is widely accepted. The advantage of neural networks is the possibility of learning the mapping from object examples to their class labels solely in a data driven way. However, to get exceptional performance, the network architecture has to be extensively tuned (number of layers, number of nodes, ...) [139]. This drawback is addressed by variants of a hybrid optimization algorithm presented in this chapter.

Other classification methods might be used instead of NNs: methods of similar complexity such as, e.g., support vector machines (SVM) or simple methods like linear classifiers, just to name a few. In this way, one could argue that classifier optimization is unnecessary. For the case of SVMs, preliminary investigations on car classification with different standard kernels (linear, polynomial, Gaussian) did indeed result in performance comparable to that of unoptimized NNs. In order to obtain better results, it seems that an optimization of SVM and kernel parameters must be performed, while an even better strategy would be to use kernels specialized to the problem classes at hand. Although it is a strong point of SVM classifiers that they allow this possibility, the design of a kernel is not always straightforward, and above all must be done manually. In contrast, the hybrid NN optimization method proposed here is largely automatic. What is more, the new kernels would presumably contain further parameters which in turn would need to be optimized.

NNs can be reduced to linear classifiers, either by choosing linear activation functions for the neurons or by removing all hidden layers and connecting the input layer directly to the output. Using the first method, the performance of linear classifiers was shown to be significantly inferior to NNs when applied to car classification.

While these investigations do not show that NNs are the best method to solve the problems at hand, they certainly suggest that NNs perform in a highly competitive way. It therefore seems justified to spend time trying to optimize them still further.

Evolutionary multi-objective optimization

Given the general problem of conflicting objectives in visual object detection and elsewhere, ways must be devised to address and resolve it. Multi-objective

optimization (MOO) considers vector-valued objective functions, where each component corresponds to one objective (e.g., speed and accuracy of a neural classifier). Such methods are capable of finding sets of trade-off solutions that give an overview of the space of possible solutions [15, 19]. A property of the found sets is that none of their elements can be said to be "better" or "worse" than another: they are "incomparable" (see later sections for a more rigorous discussion). Ideally, each incomparable solution realizes the optimum for one particular trade-off between objectives; from such sets one can select an appropriate compromise, which might not have been found by a single-objective approach.

There are recent studies that investigate domains of application and performance of evolutionary MOO applied to NN design [1, 29, 38, 45, 60, 70]. For example, Gonzalez et al. [45] evaluate the performance of different variation operators in an MOO setting for the optimization of radial basis function networks on various artificial test functions. The authors consider the number of radial basis functions for regularization and the approximation error of the network (classification error) as objective values. Both of them have to be minimized.

Fieldsend and Singh [29] suggest an evolutionary MOO algorithm for the structure optimization of four-hidden-layer NNs for stock data prediction. They use task-specific objectives, the risk and the profit, for the structure optimization of the neural regression model and compare their results to the results of a single-objective optimization approach.

Abbass [1] applies a self-adaptive evolutionary MOO algorithm with embedded gradient based weight optimization (a so called memetic or hybrid approach) to one-hidden-layer NNs trained to solve popular classification problems. The author suggests the number of hidden units (for regularization) and the approximation error of the NNs (classification accuracy) as objectives of optimization. However, a comparison to other structure optimization methods on the basis of state-of-the-art multi-objective performance assessment is missing.

Garcia-Pedrajas et al. [38] suggest an evolutionary MOO algorithm in order to tackle the credit assignment problem in cooperative coevolution of multi-layered NN-modules (sub-parts of a NN). The authors use several different objectives and compare their results to a number of other approaches applied to popular classification problems. An advanced multi-objective performance evaluation is missing.

Jin et al. [70] compare different evolutionary MOO algorithms and objective vectors applied to an artificial Ackley function in order to build ensembles of NNs from the set of Pareto optimal solutions of a trial. The approximation error of the network and different regularization terms constitute the objective vectors for optimization of NNs with one hidden layer. The authors compare the results graphically by a plot of the final results.

It is evident from these publications that the domain of evolutionary MOO of NNs is under active research, and that a number of methods exist that give excellent results in NN structure optimization. The goal of this chapter is to show that classification problems in visual object detection can profit significantly from NNs optimized by MOO. This claim is backed by applying performance

evaluation methods for multi-objective algorithms to the results.

Outline

Two NN optimization methods are applied to two data sets from real world classification problems. One method is an evolutionary MOO approach (see e.g. [60]), referred to as method EVO. In order to assess the performance of this method, the results are compared to those of the second, greedy optimization method for NNs known as magnitude-based pruning [103]. This method, referred to as PR, is evaluated in an MOO setting on an optimization problem for car classification which will be termed *car task*. The same comparison is performed on an optimization problem for face classification (denoted *face task*) [131, 132].

First, in section 7.3 the object detection and classification problems are presented. It is explained how the training databases for classification are obtained from unprocessed ("raw") video data. Those databases are used without any further reference to their origin for obtaining and improving classification functions in a purely data-driven way. Then in section 7.4, the MOO framework for structure optimization of NNs is described. Methods for comparing multi-objective optimization outcomes and the experimental setup used to derive results are given in sections 7.5 and 7.6. The results are stated in section 7.7 and discussed in section 7.8.

7.3 Optimization problems

In this section, the problems of face and car detection which give rise to the structure optimization of NNs are discussed. Both problems were described in previous publications [40, 131] and share similar system architectures: a fast initial detection produces object hypotheses (so-called *regions of interest*, ROIs) which are examined by a NN classifier, confirming or rejecting hypotheses. The initial detection stage in both cases uses heuristics which are designed for the fast detection of cars and faces. These heuristics are not learned; needless to say that they are different for cars and faces. Common to all methods for initial object detection is the requirement that the "false negative" rate (the rate of disregarded true objects) be very close to zero, whereas a moderate "false positive" rate is acceptable: it is the job of the classifier to eliminate remaining false positives, if possible. This approach embodies the requirement of capturing really *all* objects of a class; it is thus accepted that sometimes an object is detected where there really isn't one. However, usually (as shall be briefly described later) there exist additional ways of eliminating false positives beyond the scope of single-frame classification whereas there are no known methods of easily doing the reverse, that is, locating objects which are missed by the initial detection.

Input to the NN classifiers are ROIs produced by the initial detection step, their output is a decision whether the presented ROIs contain relevant objects.

The decision is based purely on the image data that is contained in the ROIs; therefore, the decision can be a function of pixel values within an ROI only. The most straightforward approach would be to present a NN classifier with the raw gray values (perhaps normalized between 0.0 and 1.0) of pixels within an ROI for learning and online classification. However, for some problems it can be profitable to perform certain transformations of the ROI data before presenting them to a classifier. The rationale behind this is twofold: on the one hand, the raw data could be transformed to a representation which is more robust to image distortions or noise, and on the other hand, some representations might facilitate classification more than others. This step is termed *feature extraction*, and the results are termed *feature sets*.

The initial detection stage will not be discussed in this chapter; instead, the classification of ROIs is emphasized. This binary classification, separating objects from non-objects, is achieved by estimating the true classification function from databases of examples. The tasks of car and face classification are now going to be described in more detail.

For both classification problems, four databases of labeled feature sets are created, termed D_{learn}, D_{val}, D_{test} and D_{ext}. Labeling means assigning a class label to each feature set indicating whether it does or does not belong to the "relevant object" class. For feature sets that are extracted online no class labels are available, of course: it is the trained classifier which has to infer those. For details about the databases please see table 7.1. The reason for this partitioning will become apparent when the learning algorithms embedded into optimization are discussed in section 7.4.1, as well as the setup of experiments in section 7.6.

In the implementation used here, the speed of a NN scales linearly with the number of connections n_{con}, which is why a small number of connections is preferable (generalization is not an issue here). The classification error $\text{CE}(D)$ is measured on some data set D. The vector-valued function $f(\text{NN}) := (n_{\text{con}}(\text{NN}), \text{CE}_{\text{NN}}(D))$ is subject to minimization. A small network size and a high classification accuracy are possibly conflicting objectives, see [3, 12].

7.3.1 Face detection data

Face detection is the inevitable first step in face recognition, aiming at localizing and extracting the regions of a video stream which contain a view of a human face. The Viisage-FaceFINDER® video surveillance system [120] automatically identifies people by their faces in a three-step process: first, regions of the video stream that contain a face are detected, then specific face models are calculated, and finally these models are compared with a database. The final face modeling and recognition is done using *Hierarchical Graph Matching* (*HGM*, [54]), which is an improvement of the *Elastic Graph Matching* method [81]. It is inspired by human vision and highly competitive to other techniques for face recognition [142]. To meet real-time constraints, the Viisage-FaceFINDER® requires very fast and accurate image classifiers within the detection unit for an optimal support of HGM.

Figure 7.1: The input to the face detection NN are 20×20 pixel feature sets showing either frontal, upright face (positive) and nonface (negative) examples. The feature extraction comprises rescaling, lighting correction, and histogram equalization.

Inputs to the face detection NN are gray-scale images which have a size of 20×20 after the feature extraction transform. The fixed-size feature sets are then classified as either containing or not containing an upright frontal face by a task specific NN [53]. Training examples therefore show either frontal, upright faces or nonfaces, see fig. 7.1. In Viisage-FaceFINDER® the NN is only a part of a sophisticated face detection module, and its main task is to support the time consuming HGM procedure with appropriate face images.

7.3.2 Car detection data

The car classification system that will be described here has been developed at our lab [40] as a component of a comprehensive software framework for Advanced Driver Assistance [10], containing modules responsible for lane detection, initial car detection, car tracking, traffic sign detection and derived tasks. For the purposes of car detection, a combination of initial detection and tracking was used prior to the development of the car classifier module: initially detected objects were tracked into adjacent frames, requiring only that they be found again sufficiently often by the initial detection modules in order to be accepted as cars. It is evident that this mechanism is less-than-perfect, because it takes several frames' time to eliminate incorrect hypotheses. Furthermore, once the initial detection produces an object which is not a car but can be tracked easily, that object will be accepted as a car (example: large rectangular traffic signs). Therefore, an approach was needed that could provide an instantaneous and accurate classification of car hypotheses. Inputs to the car detection NN are gray-scale images of arbitrary size which are transformed to a more abstract representation of size 14x14 by the feature extraction transform. The fixed-size feature sets are then classified as either containing or not containing an upright back view of a car. Training examples therefore show upright back views of cars or arbitrary views of noncars, see fig. 7.2.

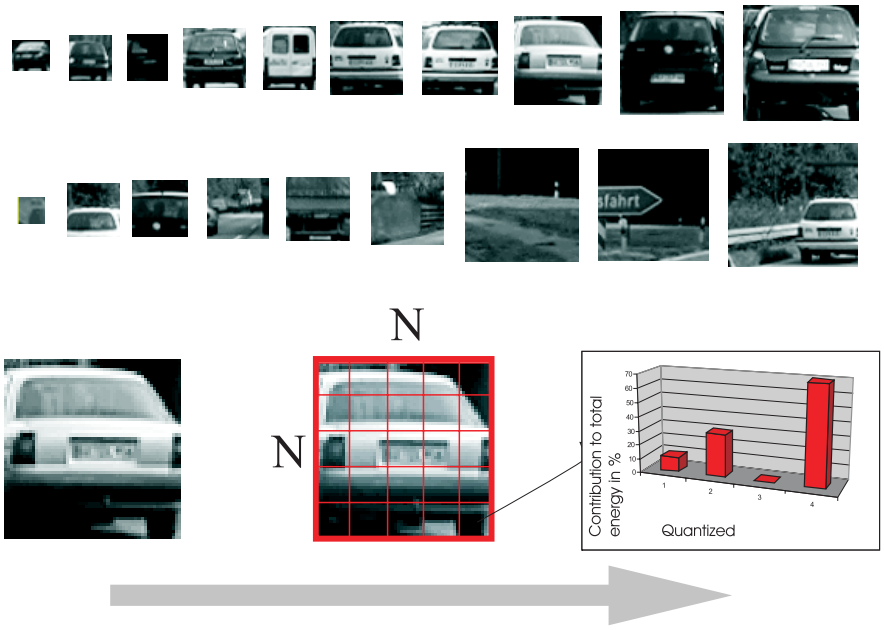


Figure 7.2: Upper half: Positive and negative training examples for the car task. Lower half: schematic depiction of feature extraction in the car task. An ROI is subdivided into NxN *receptive fields* (RF) and an identical procedure is applied to each one. Gradient orientations in each RF are coarse-grained to k values, and a total of k numbers is produced per receptive field. Each describes the contribution of gradients oriented in direction k to the total gradient strength in the RF. A choice was made using $k = 4$ and $N = 7$. A feature set thus contains N^2k values.

7.4 Optimization methods

The evolutionary optimization algorithm is presented together with multi-objective selection and magnitude-based pruning, respectively.

Optimization is performed iteratively starting from an initial population $\mathcal{P}^{(t=0)}$ of NNs. The application of an optimization method to an initial population of NNs lasting t_{max} iterations is called an optimization *trial*. An iteration t includes reproduction, structure variation, and embedded learning with some kind of cross-validation. These three steps generate the offspring population $\mathcal{O}^{(t)}$; the next parental population $\mathcal{P}^{(t+1)}$ is generated from $\mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$. A subsequent performance evaluation is used to update the *archive* $\mathcal{A}^{(t)}$. The archive represents the outcome of a trial after its completion at $t = t_{\max}$.

Table 7.1: Facts about the example data sets.

	property			usage	
data set	size(cars)	size(faces)	positives	Pruning	Evolution
D_{learn}	5000	3000	50%	Learning	Learning/Selection
D_{val}	5000	1400	50%	Crossvalidation	Crossvalidation/Selection
D_{test}	5000	2000	50%	Pareto dominance based archiving	
D_{ext}	5000	2200	50%	Estimation of generalization loss	

7.4.1 General properties of both optimization methods

In the subsequent paragraphs, each of the previously mentioned steps will be outlined, focusing on key properties common to both optimization methods.

Initialization

The comparison of results from the face task will be performed on the basis of the expert-designed 400-52-1 NN architecture, the *face reference topology*, proposed by Rowley et al. [107]. This NN has been tailored to the face detection task and has become a standard reference for NN based face detection [139]. No hidden neuron is fully connected to the input but to certain receptive fields, see below. The total number of connections amounts to 2905. This is in contrast to more than 21,000 in a fully connected NN with an equal number of hidden neurons.

In the car task, each NN is initially fully connected, has 196 input neurons, between 20 and 25 neurons in its hidden layer, one output neuron and all forward-shortcuts and bias connections in place. This architecture will be referred to as the *car reference topology*.

The NN classifiers receive a feature set computed from an ROI as input. Feature sets represent key visual properties of the ROI. In the face task the 400 numbers in the feature set correspond to the pixels of the transformed image patterns, see fig. 7.3 (right) and fig. 7.1, whereas in the car task the 196 numbers in the feature set encode higher-order visual properties as a result of advanced feature extraction methods, see fig. 7.2.

The parent population in $\mathcal{P}^{(t=0)}$ is initialized with individuals representing the *reference topologies* initialized with different small, random weight values drawn from an uniform distribution.

Reproduction and variation

Each parent creates one child per generation. First, the parent is copied. The offspring is then modified by elementary variation operators. This variation process is significantly different for pruning and the evolutionary method.

All variation operators are implemented such that their application always leads to valid NN graphs. A NN graph is considered to be *valid* if each hidden node lies on a path from an input unit to an output unit and there are no cycles. Further, the layer restriction, here set to a single hidden layer, has to be met.

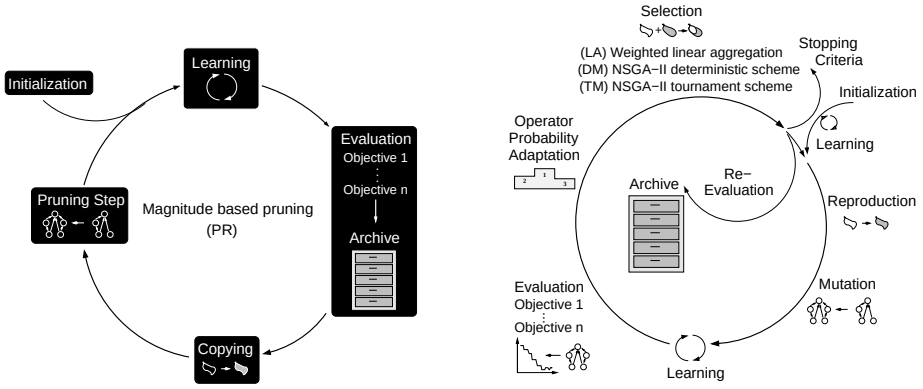


Figure 7.3: Left, a schematic overview of the pruning method PR. Right, the hybrid evolutionary algorithm EVO, see text.

Embedded learning

Let $\mathrm{MSE}_a(D)$ and $\mathrm{CE}_a(D)$ be the mean squared error and the classification error in percent on a data set D of the NN represented by individual a. Let $n_{\mathrm{con}}(a)$ denote the number of hidden neurons and weights of a, respectively. The weights of every newly generated offspring a are adapted by gradient-based optimization ("learning", "training") of $\mathrm{MSE}_a(D_{\mathrm{train}})$. An improved version of the Rprop [62, 104] algorithm is used for at most 100 epochs of training. Finally, the weight configuration with the smallest $\mathrm{MSE}_a(D_{\mathrm{train}}) + \mathrm{MSE}_a(D_{\mathrm{val}})$ encountered during training is regarded as the outcome of the training process (i.e., some kind of cross-validation comes in) and stored in the genome of the individual a in case of the face tasks (*Lamarckian inheritance*). In the car task, Lamarckian inheritance is not applied: the weights before learning are always re-initialized the same way as in the initialization of $\mathcal{P}^{(t=0)}$.

The reason for using Lamarckian inheritance in the face but not in the car task is that learning in the face task takes longer to converge. For example, the best reference topology in the car task was found after 96 epochs of learning, whereas in the face task it took 412 epochs. In the face task the accumulated knowledge of the problem by Lamarkian inheritance is necessary, since no good networks could be found during 100 epochs of embedded learning. Therefore a good starting point is helpful. In contrast, in the car task the embedded learning procedure is able to find good weight configurations during 100 epochs of weight optimization, therefore Lamarckian inheritance is unnecessary and might even result in disadvantages.

7.4.2 Magnitude-based network pruning

Pruning is a well-known reductionist method for obtaining smaller NN from larger ones by iteratively eliminating certain weights. Magnitude-based prun-

ing is a simple heuristic, but has often been reported to give satisfactory results (see [103] for a review of pruning methods); in addition, it is very easy to implement and use. Preliminary experiments using more sophisticated pruning methods [103] did not yield superior results and were therefore abandoned in favor of the most simple method: magnitude-based pruning, to which will be referred to as method PR. Please keep in mind that this does not make a general statement about the effectiveness of different pruning methods. The tested methods were Weight Elimination [127] and a sensitivity-based pruning algorithm [73], and although they produced NNs with classification accuracies comparable to method PR, the number of connections which could be eliminated was lower than when using method PR, especially in the face task. Another reason why magnitude-based pruning was finally chosen for this investigation was the intent to show that even a very primitive (in this case, *the* most primitive) optimization method can profit from multi-objective evaluation methods.

The basic loop for optimization using method PR is depicted in fig. 7.3 (left). Initialization of the first population $\mathcal{P}^{(t=0)}$ is performed as described in section 7.4.1, and reproduction simply copies the current population. Variation (here: removal of weights) is applied identically in the face and the car tasks: a percentage p of connections with the largest absolute weight is eliminated at each iteration. Learning is performed as described in section 7.4.1.

7.4.3 The evolutionary multi-objective algorithm

Evolutionary algorithms have become established methods for the design of NNs, especially for adapting their topology [60,89,140]. They are thought to be less prone to getting stuck in local optima compared to greedy algorithms like pruning or constructive methods [103, 116].

The basic optimization loop of the hybrid evolutionary algorithm is shown in fig. 7.3 (right). This scheme might be regarded as canonical evolutionary NN optimization using direct encoding and nested learning. However, there are some special features described in this section. Initialization is performed as described in section 7.4.1. It will be sketched how offsprings are created and mutated. After that, the peculiarities of the nested gradient learning procedure within the evolutionary loop will be outlined. Afterwards, the selection procedures which are considered in this work are explained. The section ends with the description of the online strategy adaptation method for adjusting the operator probabilities.

Reproduction and variation

As mentioned in section 7.4.1, each parent creates one child per generation; reproduction copies the parent population. The offspring population is then mutated by elementary variation operators. These are chosen randomly for each offspring from a set Ω of operators and are applied sequentially. The process of choosing and applying an operator is repeated $1 + x$ times, where x is an individual realization of a Poisson distributed random number with mean 1.

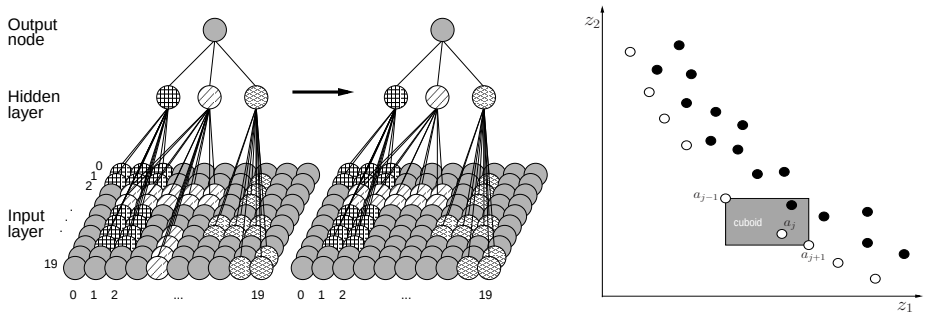


Figure 7.4: Left, scheme of the *delete-connection-RF* operator used in the *face task*. The picture also visualizes the NN input dimension and the receptive field connectivity. Right, illustration how the crowding distance $\mathrm{C}(a_j)$ [19] is computed. The black dots are the elements of M_{i+1} and the white dots belong to $\mathrm{ndom}(M_i)$, the Pareto front of M_i.

The variational operators that are to be described are similar to those introduced in [131]; The differences are due to the fact that it is a goal to reduce the number of *weights* in the work presented here, whereas the corresponding goal in [131] is to reduce the number of *nodes*.

There are four basic operators: *add-connection*, *delete-connection*, *add-node*, and *jog-weights*.

add-connection A connection is added to the NN graph. This operation is sequentially applied to the NN until the number of newly added connections amounts to 1% of the previously existent connections before *add-connection* was applied to the NN.

delete-connection This operator is inspired by *magnitude-based pruning*. The operator is rank-based as discussed by Braun [8]. The connections of the NN are sorted by the absolute value of the corresponding weights. The connection with rank number r given by

$$r := \lfloor W \cdot (\eta_{\max} - \sqrt{(\eta_{\max}^2 - 4 \cdot (\eta_{\max} - 1) \cdot u)})/(2 \cdot (\eta_{\max} - 1)) \rfloor$$

is deleted, so that connections with smaller weight have a higher probability of being removed. Here $\lfloor x \rfloor$ denotes the largest integer smaller than x, W the number of weights, and $u \sim \mathcal{U}[0,1]$ is a random variable uniformly distributed on $[0,1]$. The parameter $1 < \eta_{\max} \leq 2$ controls the influence of the rank and is set to its maximum value [129]. This operation is sequentially applied to the NN until the number of deleted connections amounts to 5% of the previously existent connections before *delete-connection* was applied.

add-node A hidden node with bias parameter is added to the NN and connected

to the output. For each input, a connection to the new node is added with probability $p_{\text{in}} = 1/16$.

jog-weights This operator adds Gaussian noise to the weights in order to push the weight configuration out of local minima and thereby to allow the gradient-based learning to explore new regions of weight space. Each weight value is varied with constant probability $p_{\text{jog}} = 0.3$ by adding normally distributed noise with expectation 0 and standard deviation $\sigma_{\text{jog}} = 0.01$.

There are 3 task-specific mutations within the face task, inspired by the concept of "receptive fields", that is, dimensions of the input space that correspond to rectangular regions of the input image, cf. fig. 7.4 (left). The RF-operators *add-RF-connection*, *delete-RF-connection*, and *add-RF-node* behave as their basic counterparts, but act on groups of connections. They consider the topology of the image plane by taking into account that "isolated" processing of pixels is rarely useful for object detection. The RF-operators are defined as follows:

add-connection-RF A valid, not yet existing connection, say from neuron i to j, is selected uniformly at random. If the source i is not an input, the connection is directly added. Otherwise, a rectangular region of the 20×20 image plane containing between 2 and $M = 100$ pixels including the one corresponding to input i is randomly chosen. Dimensions of the rectangular region are uniformly drawn, and the position is uniformly drawn from the set of possibilities fulfilling boundary conditions (i.e., no rectangular regions should exceed the input layer dimensions). Then neuron j is connected to all the inputs corresponding to the chosen image region. This operation is sequentially applied to the NN until the number of newly added connections amounts to at least 1% of the previously existent connections before *add-connection-RF* was applied.

delete-connection-RF An existing connection that can be removed, say from node i to j, is selected at random. If the source i is not an input, the connection is directly deleted. Otherwise, a decision is made whether a horizontal or vertical receptive field is deleted. Assume that a horizontal field is removed. Then *delete-connection-RF*$_x(i, j)$ is applied recursively to remove the inputs from a connected pixel row:

delete-connection-RF$_x(i, j)$ Let (i_x, i_y) be the image coordinates of the pixel corresponding to the input i. The connection from i to j is deleted. If hidden node j is also connected to the input node k corresponding to pixel $(i_x + 1, i_y)$, *delete-connection-RF*$_x(k, j)$ is applied. If j is connected to node l corresponding to $(i_x - 1, i_y)$, then the operator *delete-connection-RF*$_x(l, j)$ is called.

Deletion of a vertical receptive field (i.e., a connected pixel column) is done analogously. This operation is sequentially applied to the NN until

the number of newly deleted connections amounts to at least 5% of the previously existent connections before *delete-connection-RF* was applied.

add-node-RF A hidden node with bias connection is added and connected to the output and a receptive field as in the *add-connection-RF* operator.

There is no operator for the deletion of hidden nodes; deletion of nodes happens only when nodes no longer have any ingoing or outgoing connections. The basic operators *add-connection*, *delete-connection* and *add-node* are applied for the car task. For the face task, the operator *jog-weights* and the three task-specific mutations: *add-RF-connection*, *delete-RF-connection*, and *add-RF-node* are used in addition. Generally, weight values of new connections (produced by the operators for addition of nodes and connections) are drawn uniformly as in the first initialization of the population.

The parameter setting was not fine tuned. Instead, it was manually chosen by plausibility arguments related to the objective of reducing the number of connections in the NN. No general statement concerning the robustness of the algorithm to the choice of the parameters can be made. It would be worth investigating this topic in some future work.

Evaluations and selection in presence of multiple objectives

The goal is to find sparse NNs with high classification accuracy. That is, one tries to optimize two different objectives. The evolutionary algorithm in fig. 7.3 (right) performs advanced evolutionary MOO selection. It uses a selection method based on the Fast Non-Dominated Sorting Genetic Algorithm (*NSGA-II*) [20].

First, the elements of the decision space are mapped to n-dimensional real-valued vectors $\boldsymbol{z} = (z^1, \ldots, z^n)$ of the objective space by the *objective function* $\Phi : \mathcal{X} \to \mathbb{R}^n$. In this case, the individual a that has already finished training is mapped to the vector $\Phi(a) = \boldsymbol{z}_a = (n_{\text{hid}}(a), \text{CE}_a(D_{\text{train}} \cup D_{\text{val}}))$. Both objective components are subject to minimization. The elements of the objective space are partially ordered by the dominance relation $\prec$ ($\boldsymbol{z}$ dominates $\boldsymbol{z'}$) that is defined by

$$\boldsymbol{z} \prec \boldsymbol{z'} \in \mathbb{R}^n \quad \Leftrightarrow \quad \forall\, 1 \leq i \leq n :\ z^i \leq z'^i \ \ \wedge \ \ \exists\, 1 \leq j \leq n :\ z^j < z'^j$$

stating that vector $\boldsymbol{z}$ performs better than $\boldsymbol{z'}$ iff $\boldsymbol{z}$ is as least as good as $\boldsymbol{z'}$ in all objectives and better with respect to at least one objective. Considering a set M of n-dimensional vectors, the subset $\text{ndom}(M) \subseteq M$ consisting only of those vectors that are not dominated by any other vector of M is called the Pareto front of M. As in the *NSGA-II* (environmental) selection scheme, one first assigns to each individual $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ a rank value $\text{R}^{(t)}(a)$ based on its degree of non-domination in objective space. Let the chain of subsets M_i, $i \in \mathbb{N}$, be defined by by $M_1 \supseteq M_2 := M_1 \setminus \text{ndom}(M_1) \supseteq M_3 := M_2 \setminus \text{ndom}(M_2) \supseteq \ldots$, where $A \setminus B$ denotes the portion of the set A that is not part of set B. Then the rank operator $\text{R}^{(t)}(a)$ assigns each individual $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ the index i of the corresponding Pareto front $\text{ndom}(M_i)$ that includes the objective vector of a.

Furthermore the NSGA-II ranking takes the diversity of the population (in objective space) into account. The diversity is measured by the crowding distance $\mathrm{C}(a)$, the size of the largest cuboid (precisely the sum of its edges) in objective space enclosing the vector $\Phi(a) = \boldsymbol{z}_a$, $a \in \mathrm{ndom}(M_i)$, but no other objective vector from $\mathrm{ndom}(M_i)$, see fig. 7.4 (right). Then all individuals $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ are sorted in ascending order according to the partial order $\leq_n$ defined by

$$\begin{aligned} a_i \leq_n a_j \quad \Leftrightarrow \quad & \left(\mathrm{R}^{(t)}(a_i) < \mathrm{R}^{(t)}(a_j)\right) \text{ or} \\ & \left(\mathrm{R}^{(t)}(a_i) = \mathrm{R}^{(t)}(a_j) \wedge C(a_i) \geq C(a_j)\right). \end{aligned} \tag{7.1}$$

The selection used in the evolutionary method is chosen to be an EP-style tournament selection [30] with 5 opponents to determine the parents $\mathcal{P}^{(t+1)}$ for the next generation from $\mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$. The tournament selection is based upon the objective vector $\Phi(a) = \boldsymbol{z}_a = (n_{\mathrm{con}}(a), \mathrm{CE}_a(D_{\mathrm{train}} \cup D_{\mathrm{val}}))$, and having a "smaller" objective vector in the sense of the partial relation $\leq_n$ of eq. (7.1) increases an individual´s chance of being selected.

Search strategy adaptation: Adjusting operator probabilities

A key concept in evolutionary computation is strategy adaptation, that is, the automatic adjustment of the search strategy during the optimization process [26, 59, 63, 115]. Not all operators might be necessary at all stages of evolution. In the given case, questions such as when fine-tuning becomes more important than operating on receptive fields cannot be answered in advance. Hence, the application probabilities of the variation operators are adapted using the method from Igel and Kreutz [59], which is inspired by Davis' work [18]. The underlying assumption is that recent beneficial modifications are likely to be beneficial in the following generations. The use of search strategy adaptation is justified by the investigations described in [59, 62].

The basic operators that are actually employed in a given optimization scenario are divided into G groups, those adding connections, deleting connections, adding nodes, deleting nodes, and solely modifying weights.

Let Ω be the set of variation operators and let $p_o^{(t)}$ be the probability that $o \in \Omega$ is chosen at generation t.

The initial probabilities for operators of a single group are identical and add up to $0.25 = \frac{1}{G}$ in the face task. In the car task the initial probabilities for *add-connection* and *add-node* are set to 0.3, and the initial probability for *delete-connection* it is set to 0.4. This produces a slight bias towards deleting connections: usually with $G = 3$, the group probabilities should have been $\frac{1}{3}$.

Let $\mathcal{O}_o^{(t)}$ contain all offspring produced at generation t by an application of the operator o. The case that an offspring is produced by applying more than one operator is treated as if the offspring was generated several times, once by each operator involved. The operator probabilities are updated every τ generations. Here a choice of $\tau = 4$ is made. This period is called an adaptation cycle. The average performance achieved by the operator o over an adaptation

cycle is measured by

$$q_o^{(t,\tau)} := \sum_{i=0}^{\tau-1} \sum_{a \in \mathcal{O}_o^{(t-i)}} \max\left(0, \mathrm{B}^{(t)}(a)\right) \Big/ \sum_{i=0}^{\tau-1} \left|\mathcal{O}_o^{(t-i)}\right| \ ,$$

where $\mathrm{B}^{(t)}(a)$ represents a quality measure proportional to some kind of fitness improvement. Let

$$\mathrm{B}^{(t)}(a) := \mathrm{R}^{(t)}(\mathrm{parent}(a)) - \mathrm{R}^{(t)}(a) \ ,$$

where $\mathrm{parent}(a)$ denotes the parent of an offspring a. The operator probabilities $p_o^{(t+1)}$ are adjusted every τ generations according to eqns.

$$\tilde{p}_o^{(t+1)} := \begin{cases} \zeta \cdot q_o^{(t,\tau)} / q_{\mathrm{all}}^{(t,\tau)} & + (1-\zeta) \cdot \tilde{p}_o^{(t)} \quad \text{if } q_{\mathrm{all}}^{(t,\tau)} > 0 \\ \zeta / |\Omega| & + (1-\zeta) \cdot \tilde{p}_o^{(t)} \quad \text{otherwise} \end{cases}$$

and

$$p_o^{(t+1)} := p_{\min} \ + \ (1 - |\Omega| \cdot p_{\min}) \tilde{p}_o^{(t+1)} \Big/ \sum_{o' \in \Omega} \tilde{p}_{o'}^{(t+1)} \ .$$

The factor $q_{\mathrm{all}}^{(t,\tau)} := \sum_{o' \in \Omega} q_{o'}^{(t,\tau)}$ is used for normalization and $\tilde{p}_o^{(t+1)}$ stores the weighted average of the quality of the operator o, where the influence of previous adaptation cycles decreases exponentially. The rate of this decay is controlled by $\zeta \in (0, 1]$, which is set to $\zeta = 0.3$ in these experiments. The operator fitness $p_o^{(t+1)}$ is computed from the weighted average $\tilde{p}_o^{(t+1)}$, such that all operator probabilities sum to one and are not lower than the bound $p_{\min} < 1/|\Omega|$. Initially, $\tilde{p}_o^{(0)} = p_o^{(0)}$ for all $o \in \Omega$.

The adaptation algorithm itself has free parameters, $p_{\min}$, τ, and ζ. However, in general the number of free parameters is reduced compared to the number of parameters that are adapted and the choice of the new parameters is considerably more robust. Both τ and ξ control the speed of the adaptation; a small ξ can compensate for a small τ ($\tau = 1$ may be a reasonable choice in many applications). The adaptation adds a new quality to the algorithm as the operator probabilities can vary over time. It has been empirically shown that the operator probabilities are adapted according to different phases of the optimization process and that the performance of the structure optimization benefits from this adaptation [59, 62, 131].

7.5 Multi-objective performance assessment

Many different proposals for comparing multi-objective outcomes have been made, but common agreement as to which ones are generally preferable has not yet been reached. For a review and comparison of multi-objective performance indicators, see [77, 96, 143]. In this chapter, performance measures are used which were found to be useful in past research, and which can be easily verified by a visual inspection of the results.

To compare the multi-objective results produced by two methods, the procedures summarized in [131] are followed. For each NN, an objective vector $\boldsymbol{z}$ consists of the qualities computed singly according to all objectives. No straightforward way to compare two arbitrary objective vectors is imposed, as could be done e.g. by defining a scalar quality measure since this determines a prior trade-off between objectives. The concepts and notation from section 7.4.3 are used in what follows. The outcome of an optimization trial (for each method, a total of T trials is performed) is characterized by the Pareto front $\mathcal{A}^{(t=t_{\max})}$ using $\mathcal{A}^{(t)} := \text{ndom}(\mathcal{A}^{(t-1)} \cup \mathcal{P}^{(t)})$ with $\mathcal{A}^{(0)} = \emptyset$. In order to compare different optimization methods, ways to compare a set of trial outcomes using one method to the set produced by another method are needed. Elements of the sets are the Pareto fronts from the last-generation archive of each trial. Given two sets X and Y, *weak dominance* of sets ($X \rhd Y$) is defined as

$$X \rhd Y \text{ iff } X \neq Y \text{ and } \forall\, \boldsymbol{y} \in Y : \exists\, \boldsymbol{x} \in X : \ \boldsymbol{y} \text{ is weakly dominated by } \boldsymbol{x}. \quad (7.2)$$

The *hypervolume indicator* H_X [143] measures the percentage of objective space weakly dominated by X. Other performance indicators measure how likely the outcome X_i of trial i (using one optimization method) is to weakly dominate all trial outcomes Y_j (using another method) or to be incomparable to all of them. Let $V \subseteq \mathbb{R}^n$ be the smallest cuboid enclosing all objective vectors and m a measure. Performance indicators are defined as

$$\mathcal{P}_{X_i \rhd Y} := |\{(X_i, Y_j) : X_i \rhd Y_j,\ 1 \leq j \leq T\}| \cdot {}^1\!/_T\ , \quad (7.3)$$

$$\mathcal{P}_{X_i || Y} := |\{(X_i, Y_j) : X_i \not\rhd Y_j \wedge Y_j \not\rhd X_i,\ 1 \leq j \leq T\}| \cdot {}^1\!/_T \quad (7.4)$$

$$H_X := \left\{ m(\{z \in V | \exists z' \in X : z' \prec z\})/m(V) \right\} \in [0,1]\ . \quad (7.5)$$

In the following, A_i and B_i, $1 \leq i \leq T$ will always be used for trial outcomes using method EVO and PR respectively. For the purposes of comparison, the quantities H_{A_i}, H_{B_i}, $\mathcal{P}_{A_i||B}$, $\mathcal{P}_{B_i||A}$, $\mathcal{P}_{A_i \rhd B}$ and $\mathcal{P}_{B_i \rhd A}$, their median and median absolute deviation (mad) are calculated.

7.6 Experimental setup

The following statements hold for methods EVO and PR: All NNs have one hidden layer, activation functions are of logistic sigmoidal type. $T = 10$ trials are simulated. For each trial, a choice of $|\mathcal{P}^{(t=0)}| = 25$ is made. In the car task, each NN in $\mathcal{P}^{(t=0)}$ is fully connected, has between 20 and 25 neurons in its hidden layer and all forward-shortcuts and bias connections in place. At each iteration t, $\mathcal{P}^{(t)}$ is initialized with small random weight values between -0.05 and 0.05. This architecture is referred to as the *car reference topology*. In the face task, $\mathcal{P}^{(t=0)}$ is instantiated with 25 copies of the 400-52-1 architecture of [107], the *face reference topology*, each of which is randomly initialized like the car reference topology at $t = 0$.[2] All trials of method PR are performed for 90

[2] For $t > 0$ the weight values of the predecessor are used for initialization prior to variation.

iterations at $p = 10\%$.[3] All method EVO trials are performed for 200 iterations. The updating of the archives at each iteration is described in section 7.5.

Although cross-validation is applied when training the NNs, the evolutionary or the pruning optimization (PR) may lead to overfitting. In this case they may overfit to the patterns of $D_{\text{train}} \cup D_{\text{val}}$. Hence, the data set D_{test} is introduced in addition to choose models that generalize well and store those in the archive $\mathcal{A}^{(t)}$. That is, D_{test} is used for some kind of cross-validation of the evolutionary or pruning process. As was already explained in 7.4.3, the archive $\mathcal{A}^{(t)}$ at $t = t_{\max}$ is taken to be the final outcome of an optimization trial. The data set D_{ext}, which does not enter into the optimization at any point, is used to finally assess the performance of the members of the archive, therefore making sure the obtained results are not due to overfitting.

The car and the face reference topologies are trained 100 times for 2000 iterations using the improved Rprop learning procedure on D_{learn} and select the network a_{ref} with the smallest classification error $\text{CE}(D_{\text{val}} \cup D_{\text{test}})$. In the following, all results are normalized by the performance of a_{ref}.[4] For example, the normalized classification error of a NN a is given by $\text{CE}'_a(D) = \text{CE}_a(D)/\text{CE}_{a_{\text{ref}}}(D)$ and the normalized number of connections by $n'_{\text{con}}(a) = n_{\text{con}}(a)/n_{\text{con}}(a_{\text{ref}})$.

7.7 Results

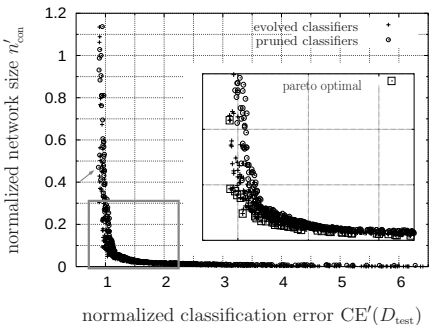
The normalized results of the car and the face task are shown in fig. 7.5 as well as in fig. 7.6 and 7.7. One perceives the surprisingly similar performance of the two methods when applied to cars. While the evolutionary method performs better, the differences are small[5] and the errors of the generated NNs are similar in similar regimes of n_{con}. In contrast, the differences between the two methods are quite pronounced when applied to the face task: here, the evolutionary MOO is clearly superior. In both tasks, the distributions of the H_{A_i} and the H_{B_j} differ in a statistically significant way.[6] All results persist when considering $(n_{\text{con}}, \text{CE}(D_{\text{ext}}))$ instead of $(n_{\text{con}}, \text{CE}(D_{\text{test}}))$, showing that no significant overfitting has occurred. An interesting observation was made when studying the results of the car task: among the smallest generated NNs are some which have no nodes at all in their hidden layer, making them essentially linear classifiers. NNs of this type correspond to trade-offs where the objective of minimizing the number of connections is important. It is very interesting to observe how the optimization tries to find the simplest possible solution for a certain desired trade-off. The performance of these linear classifier solutions is consistent with initial experiments using linear classifiers on the car task, see section 7.2.

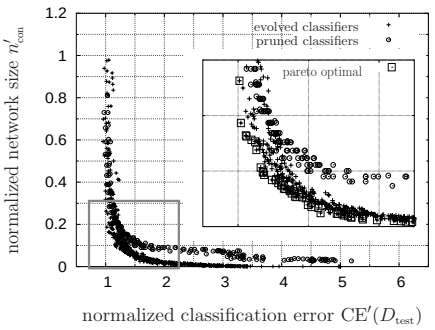
[3] No regular NN were ever produced afterwards. Reducing the pruning percentage p to the point where 200 valid NN iterations could be produced did not change results.

[4] Keep in mind that the reference topologies are not arbitrary, but tuned extensively by hand. They produce results that are highly competitive to other approaches in the literature.

[5] The actual numerical differences between indicator results can be misleading, since, e.g., the hypervolume indicator normalizes its results by the volume of the whole accessible objective space, see section 7.5.

[6] Wilcoxon Rank Sum Test, $p < 0.001$





performance indicator ($i = 1 \dots T$)	median ± mad
$\{H_{A_i}\}$	0.991±0.002
$\{H_{B_i}\}$	0.982±0.001
$\{\mathcal{P}_{A_i \rhd B}\}$	0.1±0.15
$\{\mathcal{P}_{A_i \mid\mid B}\}$	0.9±0.15
$\{\mathcal{P}_{B_i \rhd A}\}$	0.±0.0
$\{\mathcal{P}_{B_i \mid\mid A}\}$	0.8±0.15

performance indicator ($i = 1 \dots T$)	median ± mad
$\{H_{A_i}\}$	0.958±0.012
$\{H_{B_i}\}$	0.905±0.001
$\{\mathcal{P}_{A_i \rhd B}\}$	0.15±0.222
$\{\mathcal{P}_{A_i \mid\mid B}\}$	0.85±0.222
$\{\mathcal{P}_{B_i \rhd A}\}$	0.0±0.0
$\{\mathcal{P}_{B_i \mid\mid A}\}$	0.7±0.148

Figure 7.5: Left: results from the car task. Right: results from the face task. Shown on top are the unions of all trial outcomes; members of their Pareto fronts, called *meta Pareto fronts*, are shown in the magnifications. The only pruned NN in the meta Pareto front of the car task is indicated by an arrow. The performance indicators (tables at the bottom) are explained in the text.

7.8 Discussion

The result of the car task is interpreted as an indication that the problem class is intrinsically easier[7] than the face task. Therefore the simpler optimization method can yield competitive performance. For the more difficult face task, a sophisticated (here: evolutionary) optimization strategy is clearly favorable.

For the support of the interpretation about the difficulty of both tasks, one may observe that the (absolute) error $\text{CE}(D_{\text{test}})$ of the car reference topology is 3.5 times smaller than $\text{CE}(D_{\text{test}})$ of the face reference topology. As the results plainly show, optimization is unable to improve classification accuracy greatly compared to the reference topologies which constitute approximate optima in this respect. Therefore this difference in classification errors should be considered meaningful. Furthermore, optimization in the car task produced NNs without a hidden layer which nevertheless had an (absolute) classification accuracy of about 80%. This is taken as a hint that the problem is almost linearly separable and therefore can be considered "easy".

The embedding of structure optimization within an evolutionary MOO setting leads to a notable advantage compared to the single-objective formulation of the problem [131, 132]. There, a certain trade-off between partially conflict-

[7] w.r.t. the magnitude of the classification error of the best conceivable NN.

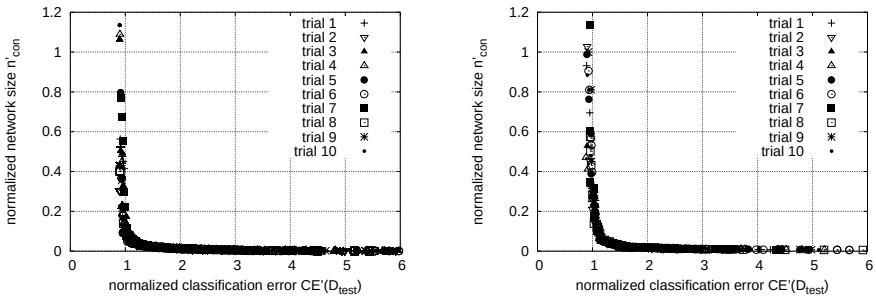


Figure 7.6: Visual comparision of optimization results for the car task. All trial outcomes (the final archives, i.e., Pareto fronts) from method EVO (left) and PR (right) are plotted into a single diagram for each method. Note the similarity between different trial outcomes of one method, making comparison to the other method easy.

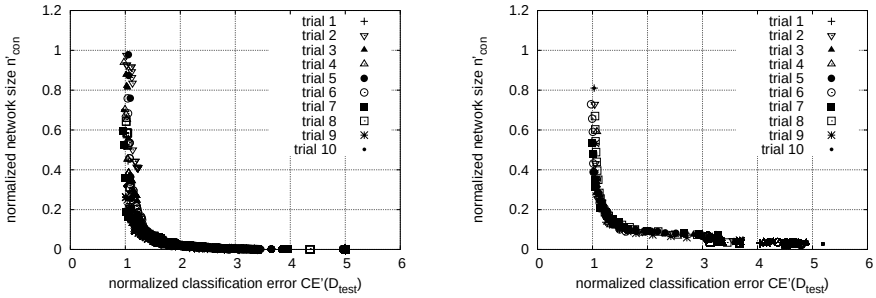


Figure 7.7: Visual comparison of optimization results for the face task. Results of method EVO are shown on the left, results of method PR on the right. For details, see fig.7.6.

ing objectives must be determined prior to search, thus disregarding certain types of solutions. In MOO, the best attainable set of incomparable solutions is generated first, and choice of specific solutions happens afterwards. It has been demonstrated that simple structure optimization heuristics like pruning can easily be incorporated in the framework of MOO. While this does not improve the optimization results themselves, one can profit from the advantages of MOO that were discussed previously.

Chapter 8

Discussion and outlook

The work presented in this thesis does not describe a single object detection system but improvements to several aspects and subproblems of object detection. The goals have been to simplify the process wherever possible by neural learning methods, and to reduce seemingly different aspects of object detection to common principles.

8.1 Applications

Some of the results described in this thesis can be used directly to improve existing object detection systems. Due to the focus of my research group, these systems are concerned with object detection in traffic scenarios. Depending on the type of traffic scenes that are considered, some assumptions about the image content can be made. For example, well-marked lane borders can always be expected in highway traffic scenes, road models can be expected to hold, and the types of behaviorally relevant objects that can conceivably appear in an image are limited. In contrast, inner-city scenes are much more complex and unpredictable, road models hold only locally, if at all, and the recognition of a variety of object types is necessary for assessing a situation correctly.

Highway traffic scenes are therefore a strongly simplified scenario, and any object detection system should work reliably in this case. However, the object detection systems developed at my research group are intended to perform well even in more difficult surroundings.

Typical objects to be detected are cars, traffic signs and lane borders. The efforts described below are already implemented or currently under development. They are briefly described in order to demonstrate that the results obtained in this thesis have direct consequences for technological applications.

Of equal importance are the possibilities for further academic research that are suggested by this thesis; they are sketched further below.

8.1.1 Trainable initial detection

Using the results from chapter 6, it is possible to construct fast initial object detectors that can be trained by examples. For this purpose, one simply uses the sparse convolutional neural networks (SCNNs) from chapter 6, keeping the network complexity (i.e., the number of independent parameters) as low as possible, thus ensuring the fastest possible execution speed. The limited network capacity may lead to misses or false detections; nevertheless, if the threshold that defines the decision of the neural network (NN) is kept low, the detection can be biased such that all incorrect classifications are false detections. This is not a problem since one can use SCNNs of increased capacity as object classifiers that classify only those ROIs that have been found by the initial detection. In this way, the increased computational effort of simulating more complex networks is only expended if it is really necessary, thus keeping the total processing time low.

It is imperative not to train the object classifiers on the same data as the initial detection SCNN. Instead, new training datasets must be constructed containing all of the previous positive examples and the false detections produced by the initial detection SCNN as negative examples.

This framework can be extended to include the simultaneous initial detection of several object classes using the feature base learning technique from chapter 6. Although each object class requires a separate object classifier, this does not increase the computational load greatly because object classifiers are only applied to ROIs provided by the initial detection. Since initial detection is performed using a common feature base, processing time is approximately independent of the number of object classes provided that feature base learning itself gives satisfactory results.

The facilitation of initial detection design is a general result of this thesis: in the following section, a specialized application is presented which directly uses the results described in chapters 4, 6 and 7.

8.1.2 Initial detection of cars and traffic signs

In order to make detection maximally robust, a project is currently pursued to integrate histogram-based initial detection and initial detection with SCNNs. For the purpose of performing initial detection using the SOE features described in chapter 4, the whole image is partitioned into receptive fields (RF) in a way that is similar to the partitioning of ROIs in chapter 4. By using NN classifiers of various fixed sizes that are optimized for speed by magnitude-based pruning,[1] the whole image can be scanned for objects very quickly. The NN classifiers are identical to those described in chapter 7.

The SOE features are additive: this means that one can compute SOE features at coarser spatial scales directly from the SOE features of finer scales.

[1] It was shown in chapter 7 that pruning is an acceptable optimization method for the car classification problem. Since traffic sign classification gives comparable classification errors, it is assumed that pruning can be applied there, too.

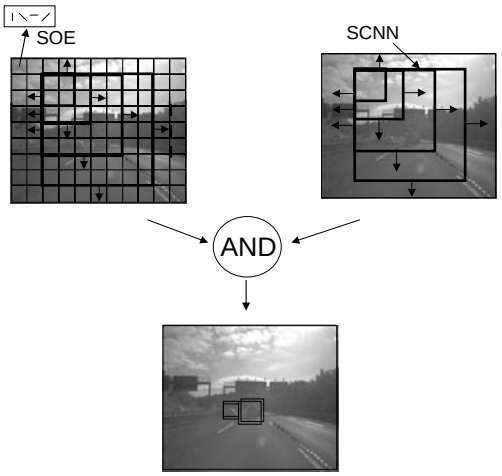


Figure 8.1: Initial detection (for clarity, only car detection is shown) by a combination of simple SCNN classifiers (upper right) discussed in chapter 6 and NN classifiers that rely on the SOE features (upper left) described in chapter 4. For the latter, the image is subdivided into small receptive fields (indicated by the grid structure) in each of which SOE features are computed. The NN classifier is applied at various scales and all positions within the image that are multiples of the receptive field size. In contrast, the SCNN classifier can be replicated over the whole image (also at several spatial scales) using finer intervals that are defined by the structure of the SCNN. The application at all possible locations is indicated by arrows in the upper row of images. The "AND"-operation symbolizes that ROIs must be detected by both methods in order to be considered

Therefore, no new information can be gained by computing SOE features at multiple scales, and the use of differently sized classifiers for multiscale object detection is thus justified.

The results of initial detection using the SOE features are compared to those obtained from the trainable initial detection using SCNNs. Only those objects that are found by both methods are considered to be detections, although a more flexible combination strategy may be favorable. Detections are then examined by object classifiers implemented by a more complex SCNN in case of cars, and by a correlation-based classifier in the case of traffic signs. Please see fig. 8.1 for an overview of the initial detection architecture. Initial results are very promising both from the point of view of detection accuracy and real-time capability.

8.1.3 Classification of lane borders

The correct detection of lane and road borders can simplify the analysis of traffic scenes considerably. On the one hand, certain image regions can be excluded

from further analysis, and on the other hand, assumptions can be made about the object classes to be expected on and off a road.

Figure 8.2: Lane detection by combining a specialized initial detection algorithm and an SCNN classifier. The initial detection produces points (indicated by small black boxes) which are conjectured to lie on a lane border. The classifier considers a small ROI around those points for its decision. Confirmed lane borders are indicated by bright boxes.

Since lane and road borders are essentially white lines, the initial detection task is comparatively simple; in contrast, real-time constraints tend to be much more restrictive because other, more time-consuming detection tasks must also be performed in real-time. Computationally demanding feature extraction is thus out of the question as well as classification with complex NNs. For this reason, SCNNs are employed for classifying ROIs; the ROIs are found by an initial detection system that was designed by an industrial partner. The advantage in this case lies in the fact that no feature extraction needs to be performed, and that the classification is fast enough to process more than 100 ROIs per image under real-time conditions (approximately 25 images per second). A typical image with correctly classified ROIs is shown in fig. 8.2.

8.2 Opportunities for further research

Beyond the mere applications that can be realized and that were described in the previous section, a number of interesting research topics suggest themselves based on results presented in this thesis. Without going into details too much, some of the possibilities are outlined.

8.2.1 Extensions of the SCNN model

The SCNN model is considerably simplified compared to the original CNN proposal [83]. This is quite intentional; however, experiments need to be conducted to find out if extensions to the SCNN model can improve its capabilities. In this

respect, the inclusion of downsampling layers (as in [83], also sketched in fig. 3.2) is worth investigating, and furthermore the possibility of shortcut connections that bypass one or several layers.

8.2.2 Research of new unsupervised learning terms for the SCNN model

The SCNN learning rule presented in chapter 6 uses an approximation to the nonlinear subspace PCA learning rule. Two issues are of interest in this respect: First of all, it would be interesting to know if other learning rules could be used. For example, it is conceivable that rules associated with independent component analysis (ICA, see, e.g., [57]) may yield superior results. On the other hand, the question is of interest how features can be generated that are correlated with the object class. Stated in another way, it should be found out if certain local features are characteristic of a certain object class *per se*, and, if so, what learning rule is needed to obtain them. Some interesting results have been published recently in this direction [49, 109] which could serve as a starting point for investigations.

8.2.3 Comparison of designed and learned feature extraction schemes

The SCNN model claims to compute meaningful and diverse object features as a part of its learning algorithm. It would be interesting to find out how the SCNN model (possibly extended using ideas from the previous section) compares to classifiers using feature extraction schemes tailored to suit their classification problems. This comparison has been done in chapter 6 for the car classification task with good results. Nevertheless, an investigation using a larger number of classification problems as benchmarks is desirable.

8.2.4 Ensemble learning with members of Pareto fronts

Multi-objective structure optimization of object classifiers as presented in chapter 7 produces no single best NN solutions but Pareto fronts of solutions. Each member of a Pareto front is optimal w.r.t. a certain trade-off between all optimization objectives. When using the objectives of speed (number of NN connections) and classification accuracy, this means that members of the Pareto front will vary from very small NNs with suboptimal classification accuracy to large NNs with optimal accuracy. Recently, ensemble learning methods have become popular [111]; it could be investigated if classification results can be improved using the AdaBoost method [111] with selected NNs from a Pareto front as an ensemble. Depending on the ensemble size, this would slow down classification; in addition, large NNs may be ensemble members, leading to a further increase in computational complexity. This need not necessarily be a problem if initial detection performs sufficiently well, which means that the number of objects to be examined by an ensemble classifier stays within reasonable limits.

AdaBoost has been shown to be beneficial w.r.t. classification accuracy and generalization ability; an empirical observation that has not yet been sufficiently explained is the fact that AdaBoost tends to be very resistant to overfitting.

8.2.5 Saliency-based object detection

The saliency map model presented in chapter 5 is currently not suitable for real-time applications but very interesting for addressing general issues of visual learning.

Figure 8.3: Example of a difficult detection task: The head of the pedestrian cannot be discerned from the background. This creates problems for detection strategies that attempt to find whole objects, in this case pedestrians.

Of particular interest to me is the issue of detecting object parts; this problem arises whenever the initial detection of whole objects is difficult. A good example is the problem of pedestrian detection (see fig. 8.3). In such situations, objects could be detected by finding parts of them first, and then inferring the position of whole objects from the positions of their parts.

Trainable saliency maps may prove to be an important tool for addressing this and related problems. The following interdependent questions seem—in my opinion—to be of special relevance:

- *Can object part categories be learned from salient object regions?*
  This can be tested by calculating the saliency map not for a whole image but only for ROIs containing training examples. By the procedure described in the discussion of chapter 5, a sequence of salient regions can be obtained in decreasing order of saliency. Then, after estimating the scale of the salient region (this should be possible by determining the scale of the strongest local feature map response), an unsupervised learning algorithm can be applied to the salient region that attempts to cluster the image content into categories. Self-organizing maps [78, 123] or similar learning methods may be of relevance at this point.

- *Can an object be robustly classified by classifying the learned parts?* Provided it is possible to learn object part categories, the learned categories could be used to characterize an object; of special interest is the question whether this is also possible when the object is partly occluded.
- *Can objects be found by detecting previously learned object parts?* Again, provided that object parts have been attributed to salient areas within objects, it is a question whether those areas are salient enough (or can be made so by learning) to be detected quickly in a cluttered image. A powerful learning algorithm for saliency maps will be crucial to these investigations.

I hereby declare this thesis finished. May you enjoy the bibliography!

Bibliography

[1] H. A. Abbass. Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, 2003.

[2] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape and motion information. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 255–260, 2005.

[3] P. L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[4] E. Baum and D. Haussler. What network size gives valid generalizations? *Neural Computation*, 1:151–160, 1989.

[5] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: a comprehensive introduction. *Natural computing*, 1(1), 3-52 2002.

[6] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.

[7] R. Bracewell. *Convolution Theorem. The Fourier Transform and Its Applications*. New York: McGraw-Hill, 3rd edition, 1999.

[8] H. Braun. *Neurale Netze: Optimierung durch Lernen und Evolution*. Springer-Verlag, 1997.

[9] L. Breimann. *Classification and regression trees (CART)*. Wadsworth, Inc. Monterey, 1984.

[10] T. Bücher, C. Curio, H. Edelbrunner, C. Igel, D. Kastrup, I. Leefken, G. Lorenz, A. Steinhage, and W. von Seelen. Image Processing and Behaviour Planning for Intelligent Vehicles. *IEEE Transactions on Industrial Electronics*, 90(1):62–75, February 2003.

[11] F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern analysis and machine intelligence*, 8:679–698, 1989.

[12] R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural networks: Backpropagation, Conjugate Gradient, and Early Stopping. In *Advances in Neural Information Processing Systems*, volume 13, pages 402–408. MIT Press, 2001.

[13] N. Christianini and J. Shawne-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[14] M. Chun and J. Wolfe. Visual attention. In E. Goldstein, editor, *Blackwell's Handbook of Perception*, chapter 9, pages 272–310. Oxford, UK: Blackwell, 2001.

[15] C. Coello Coello, D. Van Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-objective Problems*. Kluwer Academic Publishers, New York, 2002.

[16] S. Corchs and G. Deco. Feature-based attention in human visual cortex: simulation of fMRI data. *NeuroImage*, 21:36–45, 2004.

[17] S. Costa and S. Fiori. Image compression using principal component neural networks. *Image and Vision Computing*, 19(9-10):649–668, 2001.

[18] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms, ICGA'89*, pages 61–69. Morgan Kaufmann, 1989.

[19] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[20] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[21] G. Deco and E. Rolls. A neurodynamical cortical model of visual attention and invariant object recognition. *Vision Research*, 44:621–642, 2004.

[22] R. Desimone. Visual attention mediated by biased competition in extrastriate visual cortex. *Phil.Trans. R. Soc. Lond. B*, 353:1245–1255, 1998.

[23] K. Diamantaras and S. Kung. *Matrix Computations*. The John Hopkins University Press, 3rd edition, 1996.

[24] E. Dickmanns and B. Mysliwetz. Recursive 3-D road and relative ego-state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):199–213, 1992.

[25] J. Duncan. The locus of interference in the perception of simultaneous stimuli. *Psychol. Rev.*, 87:272–300, 1980.

[26] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[27] W. Einhäuser, C. Kayser, K. Körding, and P. König. Learning distinct and complementary feature-selectivities from natural colour videos. *Rev.Neurosci*, 14:43–52, 2003.

[28] D. Ferster and K. Miller. Neural mechanisms of orientation selectivity in the visual cortex. *Annual Review of Neuroscience*, 23:441–471, 2000.

[29] J. E. Fieldsend and S. Singh. Pareto evolutionary neural networks. *IEEE Transactions on Neural Networks*, 16(2):338–354, 2005.

[30] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.

[31] U. Franke and I. Kutzbach. Fast stereo based object detection for stop-and-go traffic. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 339–344, 1996.

[32] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[33] N. Friemann, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[34] S. Frintrop, G. Backer, and E. Rome. Simulating visual attention for object recognition. In *Proceedings of the Annual Meeting of the German Association for Pattern Recognition (DAGM '05)*, 2005.

[35] S. Frintrop and E. Rome. Simulating visual attention for object recognition. In *Proceedings of the Workshop on Early Cognitive Vision*, 2004.

[36] K. Fukunaga. *Introduction to statistical Pattern Recognition*. Academic press, New York, 1990.

[37] 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, November 2004.

[38] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munos-Perez. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). *Neural Networks*, 15:1259–1278, 2002.

[39] A. Gepperth. Visual object classification by sparse convolutional networks. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN) 2006*. d-side publications, 2006. submitted.

[40] A. Gepperth, J. Edelbrunner, and T. Bücher. Real-time detection of cars in video sequences. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2005)*, June 2005.

[41] A. Gepperth and S. Roth. Applications of multi-objective structure optimization. In M. Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks*, 2005.

[42] M. Girolami. *Self-Organising Neural Networks - Independent Component Analysis and Blind Source Separation*. Springer-Verlag, 1999.

[43] C. Goerick, D. Noll, and M. Werner. Artificial neural networks in real-time car detection and tracking applications. *Pattern Recognition Letters*, 17, 1996.

[44] G. Golub and C. van Loan. *Principal Component Neural Networks: Theory and Applications*. Wiley, 1996.

[45] J. Gonzalez, I. Rojas, J. Ortega, H. Pomares, J. Fernandez, and A. Diaz. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478– 1495, November 2003.

[46] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[47] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[48] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conf., Univ. Manchester*, pages 147–151, 1988.

[49] S. Hasler, H. Wersing, and E. Körner. Class-specific sparse coding for learning of object representations. In *Proc. Int. Conf. on Artif. Neur. Netw. ICANN*, pages 475–480, 2005.

[50] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[51] D. Hebb. *The organization of behaviour*. Wiley, New York, 1949.

[52] E. Hjelmas and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[53] H. M. Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, Germany, 1994.

[54] M. Hüsken, M. Brauckmann, S. Gehlen, K. Okada, and C. von der Malsburg. Evaluation of implicit 3D modeling for pose invariant face recognition. In A. K. Jain and N. K. Ratha, editors, *Defense and Security Symposium 2004: Biometric Technology for Human Identification*, volume 5404 of *Proceedings of SPIE*. The International Society for Optical Engineering, 2004.

[55] D. Huttenlocher, G. Klandermann, and W. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[56] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

[57] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*. John Wiley and Sons, Inc., 2001.

[58] C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

[59] C. Igel and M. Kreutz. Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing*, 55(1–2):347–361, 2003.

[60] C. Igel and B. Sendhoff. Evolutionary framework for the construction of diverse hybrid ensembles. In *13th European Symposium on Artificial Neural Networks (ESANN 2005)*, 2005.

[61] C. Igel and B. Sendhoff. Synergies between evolutionary and neural computation. In M. Verleysen, editor, *13th European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 241–252. Evere, Belgien: d-side publications, 2005.

[62] C. Igel, S. Wiegand, and F. Friedrichs. Evolutionary optimization of neural systems: The use of self-adptation. In M. G. de Bruin, D. H. Mache, and J. Szabados, editors, *Trends and Applications in Constructive Approximation*, number 151 in International Series of Numerical Mathematics, pages 103–123. Birkhäuser Verlag, 2005.

[63] C. Igel, S. Wiegand, and F. Friedrichs. Evolutionary optimization of neural systems: The use of self-adptation. In M. G. de Bruin, D. H. Mache, and J. Szabados, editors, *Trends and Applications in Constructive Approximation*, number 151 in International Series of Numerical Mathematics, pages 103–123. Birkhäuser Verlag, 2005.

[64] L. Itti, C. Gold, and C. Koch. Visual attention and target detection in cluttered natural scenes. *Opt. Eng.*, 40(9):1784–1793, 2001.

[65] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[66] L. Itti and C. Koch. Fea ture combination strategies for saliency-based visual attention systems. *Journal of Electronic Imaging*, 2001.

[67] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20:1254–1259, 1998.

[68] B. Jähne. *Digital image processing - Concepts, Algorithms and Scientific Applications, 3rd edition*. Springer Verlag, Berlin, 1995.

[69] B. Jähne. *Digital image processing*. Springer-Verlag, 1999.

[70] Y. Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the Congress on Evolutionary Computation (CEC'04)*, pages 1–8. IEEE Press, 2004.

[71] J. Karhunen and J. Joutsensalo. Representation and separation of signals using nonlinear PCA-type learning. *Neural Networks*, 7(1):113–127, 1994.

[72] J. Karhunen and J. Joutsensalo. Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, 8(4):549–562, 1995.

[73] E. Karnin. A simple procedure for pruning back-propagation trained networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[74] S. Kastner and G. Ungerleider. The neural basis of biased competition in human visual cortex. *Neuropsychologia*, 39:1263–1276, 2001.

[75] G. Keys. Cubic convolution interpolation for digital images. *Transactions on Acoustics, Speech and Signal Processing*, 29:1153–1160, 1981.

[76] R. Klein. Inhinition of return. *Trends in Cognitive Sciences*, 4(4):138–146, 2000.

[77] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, July 2005.

[78] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, New York, 3rd edition, 2001.

[79] V. Kotelnikow. On the transmission capacity of "ether" and wire in electrocommunications. In *Izd. Red. Upr. Svyazzi RKKA*, volume 2, 1933.

[80] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[81] M. Lades, J. C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Würtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42:301–311, 1993.

[82] J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

[83] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[84] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.

[85] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

[86] C. McAdams and J. Maunsell. Attention to both space and feature modulates neuronal responses in macaque area v4. *Journal of Neurophysiology*, 83:1751–1755, 2000.

[87] A. Messiah. *Quantum mechanics*. North-Holland, Amsterdam, 1968.

[88] P. Milner. A brief history of the hebbian learning rule. *Canadian Psychology*, February 2003.

[89] S. Nolfi. Evolution and learning in neural networks. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 415–418. MIT Press, 2nd edition, 2002.

[90] H. Nyquist. Certain topics in telegraph transmission theory. *Trans. Amer. Inst. Elect. Eng.*, 47:617–644, 1928. Reprint in: Proc.. IEEE, Vol. 90, No. 2, (Feb 2002).

[91] E. Oja. A simplified neurons model as a principal component analyzer. *J. of Mathematical biology*, 15:267–273, 1982.

[92] E. Oja. *Subspace methods of pattern recognition*. Research Studies Press, England and Wiley, USA, 1983.

[93] E. Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, pages 737–745, 1991.

[94] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *J. of Math. Analysis and Applications*, 106:69–84, 1985.

[95] E. Oja, H. Ogawa, and J. Wangviwattana. Principal component analysis by homogeneous neural networks, part i: the weighted subspace criterion. *IEICE Trans. on Information and Systems*, E75-D:366–375, 1991.

[96] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'03)*, pages 1053–1060. IEEE Press, 2003.

[97] B. Olshausen and D. Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7:333–339, 1996.

[98] M. Oren, C. Papageorgiou, P. Sinha, T. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition, Puerto Rico*, pages pp. 193–199, 1997.

[99] M. Oren, C. Papageorgiou, P. Sinha, T. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition, Puerto Rico*, pages pp. 193–199, 1997.

[100] D. Parkhurst, K. Law, and E. Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002.

[101] P. Penev and J. Atick. Local feature analysis: A general statistical theory for object recognition. *Network: computation in neural systems*, 7(3):477–500, 1996.

[102] R. Peters, A. Iyer, L. Itti, and C. Koch. Components of bottom-up gaze allocation in natural images. *Vision Research*, 45(18), 2397-2416 2005.

[103] R. D. Reed and R. J. Marks II. *Neural Smithing*. MIT Press, 1999.

[104] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(5):265–278, 1994.

[105] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[106] A. Rosenfeld and A. Kak. *Digital picture processing*. Academic Press, San Diego, 1982.

[107] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[108] A. Rybak, A. Gusakova, L. Golovan, and N. Shevtsova. A model of attention-guided visual perception and recognition. *Vision research*, 38:2387–2400, 1998.

[109] Y. Sakaguchi, S. Ozawa, and M. Kotani. Feature extraction using supervised independent component analysis by maximizing class distance. In *Proceedings of the 9th International Conference on Neural Information Processing ICONIP*, pages 2502– 2506, 2002.

[110] T. Sanger. Optimal unsupervised learning in a single-layered linear feedforward network. *Neural Networks*, 2:459–473, 1989.

[111] R. Schapire and Y. Freund. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330, 1997.

[112] H. Schneidermann and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, 2000.

[113] C. Shannon. Communication in the presence of noise. In *Proc. IRE*, volume 37, 1949. Reprint in: Proc.. IEEE, Vol. 86, No. 2, (Feb 1998).

[114] A. Shashua, Y. Gdalyahu, and G. Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 1–6, 2004.

[115] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.

[116] A. Stahlberger and M. Riedmiller. Fast network pruning and feature extraction by using the unit-OBS algorithm. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 655–661. The MIT Press, 1997.

[117] Z. Sun, G. Bebis, and R. Miller. Object detection using feature subset selection. *Pattern recognition*, 37:2165–2176, 2004.

[118] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata. Pedestrian detection using convolutional neural networks. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 224–229, 2005.

[119] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520, June 1996.

[120] Viisage Technology AG. `http://www.viisage.com`.

[121] 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*, 2001.

[122] P. Viola, M. Jones, and D. Snow. Pedestrian detection using patterns of motion and appearance. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[123] C. von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14:85–100, 1973.

[124] D. Walther, L. Itti, M. Riesenhuber, T. Poggio, and C. Koch. Attentional selection for object recognition - a gentle way. In *Proceedings of the workshop on biologically motivated computer vision*, volume 2525, pages 472–479, 2002.

[125] B. Wandell. *Foundations of vision*. Sunderland, Mass.: Sinauer Associates, 1995.

[126] A. Webb. *Statistical Pattern Recognition*. Wiley, 2nd edition, 2002.

[127] A. S. Weigend, D. E. Rumelhart, and B. Huberman. Back-propagation, weight elimination and time series prediction. In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 105–116. Morgan Kaufmann, Sant Matteo, 1991.

[128] H. Wersing and E. Körner. Unsupervised learning of combination features for hierarchical recognition models. In *Proceedings of the ICANN*, 2002.

[129] L. D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms, ICGA'89*, pages 116–121. Morgan Kaufmann, 1989.

[130] J. Whittaker. The fourier theory of the cardinal functions. In *Proc. Edinburgh Math. Soc.*, volume 1, 1929.

[131] S. Wiegand, C. Igel, and U. Handmann. Evolutionary multi-objective optimisation of neural networks for face detection. *International Journal of Computational Intelligence and Applications*, 4(3):237–253, 2004.

[132] S. Wiegand, C. Igel, and U. Handmann. Evolutionary optimization of neural networks for face detection. In M. Verleysen, editor, *12th European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 139–144. Evere, Belgium: d-side publications, 2004.

[133] C. Wöhler and J. K. Anlauf. Real-time object recognition on image sequences with the adaptable time delay neural network algorithm - applications for autonomous vehicles. *Image and Vision Computing*, 19(9-10):593–618, 2001.

[134] J. Wolfe. Visual search. In H. Pashler, editor, *Attention*. London UK: University College London Press, 1995.

[135] J. Wolfe. Extending guided search: why guided search needs a preattentive "item map". In A. Kramer, M. Coles, and G. Logan, editors, *Converging operations in the study of visual attention*, pages 247–270. Washington DC: American Psychological Association, 1996.

[136] J. Wolfe. Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238, 1996.

[137] J. M. Wolfe. Visual search. In H. D. Pashler, editor, *Attention*. London UK: University College London Press, 1998.

[138] L. Xu. Least mean square error reconstruction principle for self-organizing neural nets. *Neural Networks*, 6:627–648, 1993.

[139] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[140] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[141] G. P. Zhang. Neural Networks for Classification: A Survey. *IEEE Transactions on System, Man, and Cybernetics – Part C*, 30(4):451 – 462, 2000.

[142] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399 – 458, 2003.

[143] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Errata

- In fig. 1.4 on p.21, the labels for the x and y axis should be "Spatial frequency" and "Transfer function"
- In fig. 1.5 (left) on p.22, the z axis label should read "Convolution filter" instead of "Transfer function"
- Figs. 8.1 and 8.2 on pages 106 and 107 were badly reproduced during the printing process, so that important details cannot be perceived. Correctly, they should look loke this: