Multi-objective neural network optimization for visual object detection

Stefan Roth[1], Alexander Gepperth[2], and Christian Igel[3]

[1] stefan.roth@neuroinformatik.rub.de
[2] alexander.gepperth@neuroinformatik.rub.de
[3] christian.igel@neuroinformatik.rub.de

Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

In real-time computer vision, there is a need for classifiers that detect patterns fast and reliably. We apply multi-objective optimization (MOO) to the design of feed-forward neural networks for real-world object recognition tasks, where computational complexity and accuracy define partially conflicting objectives. Evolutionary structure optimization and pruning are compared for the adaptation of the network topology. In addition, the results of MOO are contrasted to those of a single-objective evolutionary algorithm. As a part of the evolutionary algorithm, the automatic adaptation of operator probabilities in MOO is described.

1 Introduction

When speaking of real-world object detection we usually refer to object detection tasks having some application in the commercial domain. These tasks must be solved well enough to allow their application in products, where error tolerances are usually very restrictive. In the ideal case, the detection accuracy should be comparable or superior to that of humans. When considering existing applications of real-world object detection systems it is clear that imperfect performance can lead to serious (possibly fatal, e.g., in automotive applications) problems. This imposes 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 constraints 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 domains of advanced driver assistance and biometric systems. Advanced driver assistance systems typically require the detection of pedestrians [41, 44], vehicles [49, 32, 21], lane borders [13], or traffic signs [2] to ensure that the "intelligent vehicles" can construct an adequately complete representation of their surroundings and (possibly) take the appropriate actions. Within biometric systems, face recognition is of particular interest. Automatic face recognition 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 or an image that contain a view of a human face. Overviews of examples, problems, and approaches in the research domain of face detection can be found in [24, 51].

Thus, the problems of detecting cars and human faces are important examples of current research in visual object detection. Usually, computer vision architectures are broadly motivated by biological visual search strategies [50, 8]: 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. The scope of this contribution is on the classification of hypotheses, that is, on the decision whether a given object hypothesis actually corresponds to a relevant object type. Based on previous work of the authors, the problems of car and face detection are discussed in-depth.

Neural classifiers

We address the task of optimizing the weights and the structure of feed-forward neural networks (*FFNNs*) used for face and car classification in the Viisage-FaceFINDER® system and the car detection system described in [21], respectively. 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 enhancing the accuracy of the FFNN classifiers. It is unreasonable to expect that these two requirements can be achieved independently and simultaneously.

Feed-forward neural networks have proven to be powerful tools in pattern recognition [53]. Especially (but not only) in the domain of face detection the competitiveness of FFNNs is widely accepted. As stated in a recent survey "The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. However, one drawback is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance" [51]. This drawback is addressed by variants of a hybrid optimization algorithm presented in this article.

Evolutionary multi-objective optimization

Given the general problem of conflicting objectives in visual object detection, ways must be devised to address and resolve it. Advanced evolutionary multi-objective optimization (EMO) considers vector-valued objective functions, where each component corresponds to one objective (e.g., speed or accuracy of a neural classifier). Such methods are capable of finding sets of trade-off solutions, none of which can be said to be superior to another without an explicit weighting of single objectives [11, 9]. From such a set 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 EMO applied to FFNN design [29, 23, 16, 1, 20, 31]. It is evident from these publications that the EMO of FFNNs is under active research, and that a number of methods exist that give excellent results in FFNN structure optimization. The goal of this contribution is to show that visual object detection can profit significantly from FFNNs optimized by EMO.

Outline

In this article we summarize the results of our work in the domain of EMO for FFNN structure optimization [47, 48, 22]. We present variants of our self-adaptive, hybrid EMO and demonstrate the performance in contrast to the performance of a greedy optimization method for FFNN design known as magnitude-based pruning [38]. We evaluate our methods on an optimization problem for car classification, which will be termed *car task*, and on a face detection problem denoted *face task*.

First, in section 2 we present the object detection tasks that embed the considered classification problems. We explain how the training data for classification are obtained from unprocessed ("raw") video data. In section 3, the EMO framework for structure optimization of FFNNs is described. State-of-the-art methods for comparing multi-objective optimization outcomes and the experimental setup used to derive results are given in sections 4 and 5. The results are stated in section 6 and discussed in section 7.

2 Optimization problems

In this section, we describe the optimization problems of improving FFNNs for face and car detection. Both detection tasks were introduced in previous publications [21, 47] and share similar system architectures: a fast initial detection produces object hypotheses (so-called *regions of interest*, ROIs), which are examined by an FFNN classifier confirming or rejecting hypotheses. The initial detection stage in both cases uses heuristics designed for the fast detection of cars and faces, respectively. These heuristics are not learned but designed and 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 aims at capturing *all* objects of a class; it is accepted that sometimes an object is detected where there is none. 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 FFNN classifiers are ROIs produced by the initial detection step, their output is a decision whether the presented ROIs contain relevant objects or not. The decision is based purely on the image data that are 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 the raw gray values (perhaps normalized between 0.0 and 1.0) of pixels within an ROI to an FFNN classifier for learning and online classification. However, for many problems it is profitable to perform certain transformations of the ROI data before presenting them to a classifier. Some representations facilitate classification more than others, for example, the raw data could be transformed to a representation which is more robust to image distortions or noise.

We will not discuss the initial detection stage in this contribution but emphasize the classification of ROIs instead. This binary classification, separating objects from non-objects, is achieved by estimating the true classification function from sample data. We are now going to describe the tasks of car and face classification in more detail, focusing on the process of obtaining training and online examples (also termed *feature sets*) from the raw image data, a process which is frequently termed *feature extraction*.

For both classification problems, we create four data sets of labeled examples termed D_{learn}, D_{val}, D_{test}, and D_{ext}. Labeling means assigning a class label to each example indicating whether it belongs to the "relevant object" class. For details about the data sets please see Tab. 1. The reason for this partitioning will become apparent when we discuss the experimental setup in section 5.

2.1 Face detection data

The Viisage-FaceFINDER® video surveillance system [45] automatically identifies people by their faces in a three-step process. First, regions of the video

Table 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	

stream that contain a face are detected, then specific face models are calculated, and finally these models are compared to a database. The final face modeling and recognition is done using *Hierarchical Graph Matching* (*HGM*, [26]), which is an improvement of the *Elastic Graph Matching* method [34]. It is inspired by human vision and highly competitive to other techniques for face recognition [54]. 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.

Inputs to the face detection FFNN are preprocessed 20×20 pixel grayscale images, which show either frontal, upright faces or nonfaces, see Fig. 1. In the preprocessing step, different biologically motivated cues are fused to cluster the given images into regions of high and low significance (i.e., ROIs and background). The preprocessing comprises further rescaling, lighting correction, and histogram equalization. The fixed-size ROIs are then classified as either containing or not containing an upright frontal face by a task specific FFNN [25].

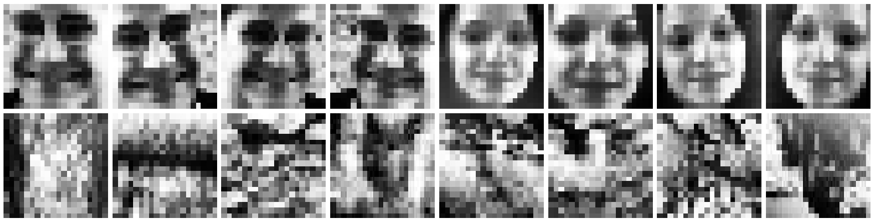


Fig. 1. Input to the face detection FFNN are preprocessed 20×20 pixel grayscale images showing either frontal, upright face (positive) and nonface (negative) examples. The preprocessing comprises rescaling, lighting correction, and histogram equalization.

The assumption of fixed-size ROIs as input to the classifier meets the realistic application scenario for the FFNN in the Viisage-FaceFINDER® system, although in the survey on face detection by Hjelmas and Low such input patterns are regarded as unrealistic for real world face detection [24]. In Viisage-FaceFINDER® the FFNN 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.

We use the face detection problem for two slightly different test scenarios. When we refer to the *face task* in the following, we mean both of them. In the first test scenario $\boxed{1}$, see section 5.1—comparison of advanced EMO-selection vs. selection of multiple objectives via linear aggregation—the goal is to reduce the number of hidden nodes of the detection FFNN. This is because in the hardware-friendly implementation of the face detection FFNN within Viisage-FaceFINDER® the speed of the classification scales approximately linearly with the number of hidden neurons and not with the number of connections. With every hidden neuron that is saved the detection costs are reduced by approximately one percentage point. In the second test scenario $\boxed{2}$, refer to section 5.2—comparison of pruning vs. evolutionary algorithms in a multi-objective framework—, the first objective is to reduce the amount of connections in the network. This is due to another FFNN implementation where the processing speed scales approximately linearly with the number of connections as well as nodes. In both test scenarios, the second objective is identical: improving classification accuracy. A small network size and a high classification accuracy are possibly conflicting objectives (note that “the smaller the network the better the generalization” does not hold in general [3, 7]).

2.2 Car detection data

The car classification system we are going to describe here has been developed at our lab [21] as a component of a comprehensive software framework for Advanced Driver Assistance [6] 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 are 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 that 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.

The transformed data computed from a car ROI are based on image gradients (and derived quantities) only: raw pixel values are not considered at all. Gradient information is very useful since it is usually quite robust w.r.t. changes in lighting conditions. From the two gradient images (gradients taken in x- and y-direction) denoted $G_x(x, y), G_y(x, y)$, an *energy image* is

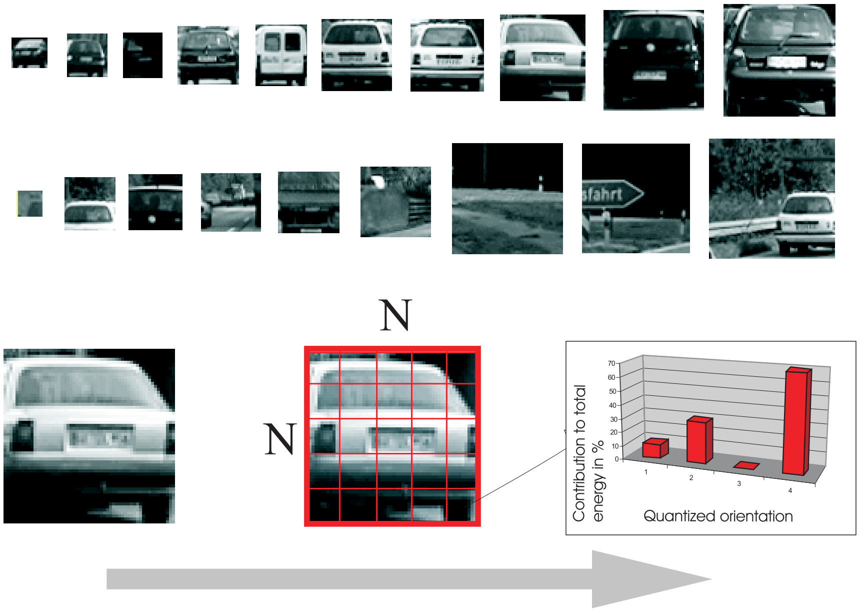


Fig. 2. Upper half: Positive and negative training examples for the car task. Since we are using a scale-invariant feature extraction method, examples are not rescaled and can vary significantly in size. Lower half: schematic depiction of a feature vector generated from a single training example, see section 2.2 for an explanation of the quantities used here. An ROI is subdivided into N × N *receptive fields* (RF) and an identical procedure is applied to each one. Calculated orientations in each RF are quantized to k values. For each of the k possible orientation values $i \in [1, .., k]$, and RF pixels (x, y), the quantity $\nu_i = \frac{1}{E_{RF}} \sum_{(x,y)} \chi(x, y)$ is computed, where $\chi(x, y) = E(x, y)$ if $A(x, y) = i$, and 0 otherwise. E_{RF} is obtained by summing up $E(x, y)$ over the whole RF. Thus, k numbers are produced per receptive field that describe the contribution a single orientation makes to the total summed-up edge energy in a receptive field. We choose $k = 4$ and N = 7. A feature vector thus contains N^2k values.

derived by the formula $E(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$. Wherever $E(x, y)$ exceeds a certain threshold, we compute the value of the *angle image* to $A(x, y) = \arctan \frac{G_y(x,y)}{G_x(x,y)}$ and zero otherwise. Due to practical considerations, the value $A(x, y) \in [0, \pi]$ is often quantized to k orientation directions. The details of the extraction process are described in Fig. 2 (upper half).

Several comments are in order: The feature extraction process is designed in the described way in order to incorporate scale, translation, and lighting invariance already in the feature set, at least to a certain extent. It is a simplified implementation of orientation-selective processing, which has been shown to be abundant in biological vision systems [15]. Many successful computational models for detection and classification rely heavily on orientation-selective feature extraction (examples are wavelet representation like Gabor- or steerable filters [36, 19]), underlining the effectiveness of this approach. In addition, a favorable processing speed is ensured by using only gradient information since image gradients can be computed very efficiently.

In the following we refer to the problem of optimizing an FFNN car classifier on the basis of a collection of feature vectors as the *car task*. The car task is only considered in the second test scenario $\boxed{2}$, see section 5.2—comparison of pruning vs. evolutionary algorithms in a multi-objective framework. Hence, in the car task the objectives for optimization are the classification error and the number of connections as described in section 2.1.

3 Optimization methods

We present our evolutionary optimization algorithm with variants of multi-objective selection and magnitude-based pruning, respectively. Optimization is performed iteratively starting from an initial population $\mathcal{P}^{(t=0)}$ of FFNNs. An iteration t includes reproduction, structure variation, and embedded learning with some kind of cross-validation (CV). These three steps generate the offspring population $\mathcal{O}^{(t)}$. In the evolutionary algorithms, these offspring are evaluated and the individuals for the new parent population $\mathcal{P}^{(t+1)}$ are selected from $\mathcal{O}^{(t)}$ and $\mathcal{P}^{(t)}$. For every evaluation an individual of the decision space $\mathcal{X}$ (the genotype space—the space of the encoded FFNNs) is mapped to an n-dimensional vector of objective space by a quality function $\Phi : \mathcal{X} \to \mathbb{R}^n$.

3.1 General properties of all optimization methods

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

Initialization

The comparison of our results using the face task will be performed on the basis of the expert-designed 400-52-1 FFNN architecture, the *face reference topology*, proposed by Rowley et al. [40]. This FFNN has been tailored to the face detection task and has become a standard reference for FFNN based face detection [51]. 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 FFNN with an equal number of hidden neurons.

In the car task, each FFNN is initially fully connected, has 196 input neurons, between 20 and 25 neurons in its hidden layer, one output neuron and all

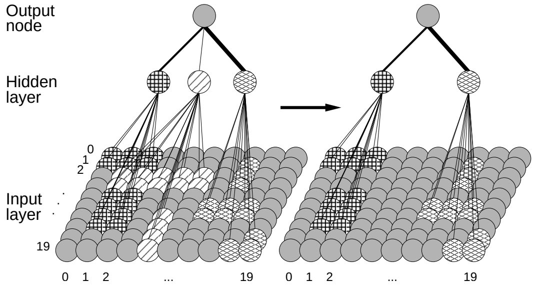


Fig. 3. Visualization of the *delete-node* operator within the test scenario [1]. The line widths indicate the magnitude of the corresponding weight values. The picture also visualizes the FFNN input dimension and the receptive field connectivity.

forward-shortcuts and bias connections in place. We refer to this architecture as the *car reference topology*.

As input, the FFNN classifiers receive a feature set, representing key visual properties of the ROI which it is computed from. In the face tasks the 400 numbers in the feature set correspond to the pixels of the preprocessed image patterns, see Fig. 1 and Fig. 3, 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. 2 (lower half).

We create the parent individuals in $\mathcal{P}^{(t=0)}$ as copies of the *reference topologies*, which are initialized with different, small random weight values.

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 methods.

All variation operators are implemented such that their application always leads to valid FFNN graphs. An FFNN 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.

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 FFNN represented by individual a. 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 [27, 39] algorithm is used for at most 100 epochs of training. Finally, the weight configuration with the smallest $\mathrm{MSE}_a(D_{\mathrm{train}}) + \mathrm{MSE}_a(D_{\mathrm{val}})$

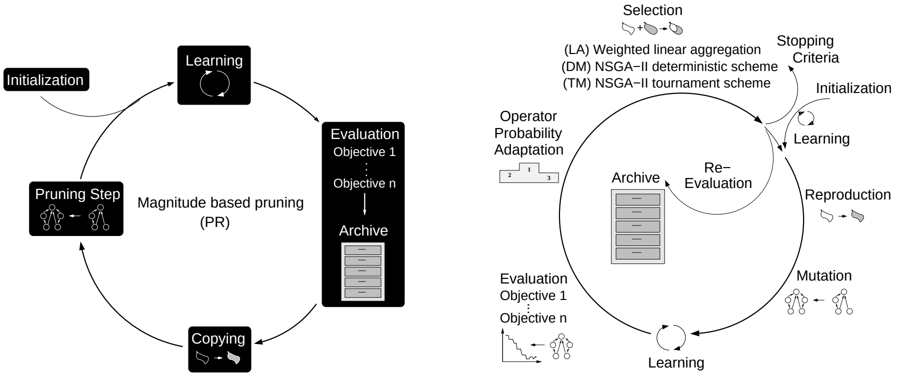


Fig. 4. Left, a schematic overview of the pruning method PR. Right, the hybrid evolutionary algorithm in conjunction with three different selection variants LA, DM and TM, see text.

encountered during training is regarded as the outcome of the training process (i.e., some kind of CV 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 $\mathcal{P}^{(t=0)}$.

Dominance based Archiving

For the pruning method and all evolutionary variants, another performance evaluation of the parental individuals $a \in \mathcal{P}^{(t)}$ is used to update the *external archive* $\mathcal{A}^{(t)}$ at iteration t. This performance evaluation is based on a second mapping of the individuals a to n-dimensional objective vectors $\boldsymbol{z} \in \mathbb{R}^n$. The external archive represents the outcome of a trial after its completion at $t = t_{\max}$. In the following let $n_{\text{hid}}(a)$ and $n_{\text{con}}(a)$ denote the number of hidden neurons and weights of the individual $a \in \mathcal{P}^{(t)}$, respectively.

3.2 Magnitude-based network pruning

Pruning is a well-known reductionist method for obtaining smaller FFNNs from larger ones by iteratively eliminating certain weights. Magnitude-based pruning is a simple heuristic, but has often been reported to give satisfactory results (see [38] for a review of pruning methods). In addition, it is very easy to implement and use. Preliminary experiments using more sophisticated pruning methods [38] did not yield superior results and were therefore abandoned in favor of the most simple method: magnitude-based pruning which we will refer to as method PR. The basic loop for optimization using PR is depicted in Fig. 4 (left). Initialization of the first population $\mathcal{P}^{(t=0)}$ is performed as described in section 3.1, and reproduction simply copies the current population.

Variation (here: weight elimination) is applied identically in the face and the car tasks: a percentage p of connections with the smallest absolute weight is eliminated at each iteration. Learning is performed as described in section 3.1.

3.3 The evolutionary multi-objective algorithm

Evolutionary algorithms have become established methods for the design of FFNNs, especially for adapting their topology [35, 52, 29]. They are thought to be less prone to getting stuck in local optima compared to greedy algorithms like pruning or constructive methods [38, 43].

The basic optimization loop of our hybrid evolutionary algorithm is shown in Fig. 4 (right). This scheme might be regarded as canonical evolutionary FFNN optimization using direct encoding and nested learning. However, there are some special features described in this section.

Initialization is performed as described in section 3.1. We will sketch how offspring are created and mutated. After that, we outline the peculiarities of the nested gradient-based learning procedure within the evolutionary loop. Then we highlight the three different approaches to selection which are considered in this work. The section ends with the description of the online strategy adaptation method for adjusting operator application probabilities.

Reproduction and variation

As mentioned in section 3.1, 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.

We need to distinguish the operators for the two test scenarios $\boxed{1}$ and $\boxed{2}$, due to the requirement of reducing the number of nodes instead of connections. In the first test scenario $\boxed{1}$ there are 5 basic operators: *add-connection*, *delete-connection*, *add-node*, *delete-node*, and *jog-weights*:

add-connection A connection is added to the FFNN graph.

delete-connection This operator is inspired by *magnitude-based pruning*. The operator is rank-based as discussed by Braun [5]. The connections of the FFNN are sorted by the absolute value of the corresponding weights. The connection with rank number r given by

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

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 [46].

add-node A hidden node with bias parameter is added to the FFNN and connected to the output. For each input, a connection to the new node is added with probability $p_{\text{in}} = 1/16$.

delete-node In this rank-based node deletion operator, the hidden nodes are ordered according to their maximum output weight. The maximum output weight of a node i is given by $\max_j |w_{ji}|$, where w_{ji} is the weight of the connection from node i to node j. The nodes are selected based on eq. (1), such that nodes with smaller maximum output weight values have a higher probability of deletion. If node k is deleted, all connections to or from k are removed, cf. Fig. 3.

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 value 0 and standard deviation $\sigma_{\text{jog}} = 0.01$.

In addition to the 5 basic operators, there are 3 task-specific mutations within scenario 1 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. 3. 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. Then neuron j is connected to all the inputs corresponding to the chosen image region.

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.

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.

In the second test scenario 2 we use different operators for the insertion and deletion of connections than in scenario 1. Furthermore, 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. We apply the basic operators *add-connection-P*, *delete-connection-P* and *add-node* for the car task. For the face task, we additionally use the operator *jog-weights* and three task-specific mutations: *add-RF-connection-P*, *delete-RF-connection-P*, and *add-RF-node*. The new mutation operators are defined as follows:

add-connection-P The operator *add-connection* is sequentially applied to the FFNN until the number of newly added connections amounts to 1% of the previously existent connections before *add-connection-P* was applied.

delete-connection-P The operator *delete-connection* is sequentially applied to the FFNN until the number of deleted connections amounts to 5% of the previously existent connections before *delete-connection-P* was applied.

add-connection-RF-P The operator *add-connection-RF* is sequentially applied to the FFNN until the number of newly added connections amounts to at least 1% of the previously existent connections before *add-connection-RF-P* was applied.

delete-connection-RF-P The operator *delete-connection-RF* is sequentially applied to the FFNN until the number of deleted connections amounts to at least 5% of the previously existent connections before the operator *delete-connection-RF-P* was applied.

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.

Embedded learning

A peculiarity of the evolutionary methods is the fact that training can stop earlier due to the *generalization loss* criterion GL_α as described by Prechelt [37]. The generalization loss is computed on D_{val} for $\alpha = 5$. This is done in order to reduce the computational cost of the optimization.

Evaluations and selection in presence of multiple objectives

We are looking for sparse FFNNs with high classification accuracy. That is, we try to optimize two different objectives. There are several ways of dealing with multiple goals, and we will describe three of them in the following.

LA—*Linearly aggregated objectives are subject to selection.*

In the first case of the test scenario $\boxed{1}$ the algorithm in Fig. 4 (right) uses a scalar fitness $a \mapsto \Phi(a) \in \mathbb{R}$ for any individual a given by the weighted linear aggregation

$$\begin{aligned}\Phi(a) := \quad & \gamma_{\mathrm{CE}} \cdot \mathrm{CE}_a^{(t)}(D_{\mathrm{train}} \cup D_{\mathrm{val}}) && + \mathrm{MSE}_a^{(t)}(D_{\mathrm{train}} \cup D_{\mathrm{val}}) \\ & + \gamma_{\mathrm{hid}} \cdot n_{\mathrm{hid}}^{(t)}(a) && + \gamma_{\mathrm{con}} \cdot n_{\mathrm{con}}^{(t)}(a)\end{aligned} \tag{2}$$

that is to be minimized. The weighting factors are chosen such that typically $\gamma_{\mathrm{CE}} \cdot \mathrm{CE}_a^{(t)}(D_{\mathrm{train}} \cup D_{\mathrm{val}}) \gg \gamma_{\mathrm{hid}} \cdot n_{\mathrm{hid}}^{(t)}(a) \approx \gamma_{\mathrm{con}} \cdot n_{\mathrm{con}}^{(t)}(a) \gg \mathrm{MSE}_a^{(t)}(D_{\mathrm{train}} \cup D_{\mathrm{val}})$ holds. Note that in test scenario $\boxed{1}$ we tolerate an increase in the number of connections as long the number of neurons decreases.

Based on the fitness Φ, EP-style tournament selection [17] with 5 opponents is applied to determine the parents $\mathcal{P}^{(t+1)}$ for the next generation from $\mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$. We refer to the described selection method as *linearly-aggregated selection* LA and identify the complete algorithm with this selection scheme throughout this article.

DM—*Vector-valued objectives are subject to deterministic selection.*

In the second case of the test scenario $\boxed{1}$, the evolutionary algorithm in Fig. 4 (right) performs advanced EMO selection. It uses a selection method based on the Fast Non-Dominated Sorting Genetic Algorithm (*NSGA-II*) [12].

We first map the elements of the decision space to n-dimensional real-valued vectors $\boldsymbol{z} = (z_1, \dots, z_n)$ of the objective space by the objective function $\Phi : \mathcal{X} \to \mathbb{R}^n$. In our case we map the individual a that has already finished training to the vector $\Phi(a) = \boldsymbol{z}_a = (n_{\mathrm{hid}}(a), \mathrm{CE}_a(D_{\mathrm{train}} \cup D_{\mathrm{val}}))$. 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 $\mathrm{ndom}(M) \subseteq M$ 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, we first assign a rank value $\mathrm{R}^{(t)}(a)$ to each individual $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ based on its

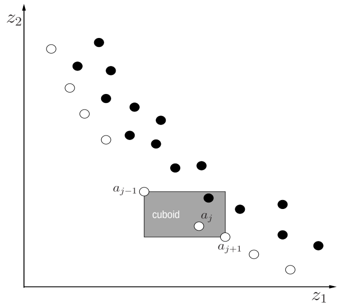


Fig. 5. Illustration how the crowding distance $\mathrm{C}(a_j)$ [11] is computed. The black dots are the elements of M_{i+1} and the white dots of belong to the Pareto front $\mathrm{ndom}(M_i)$.

degree of non-domination in objective space. We define the chain of subsets M_i, $i \in \mathbb{N}$, by $M_1 \supseteq M_2 := M_1 \setminus \mathrm{ndom}(M_1) \supseteq M_3 := M_2 \setminus \mathrm{ndom}(M_2) \supseteq \dots$, where $A \setminus B$ denotes the portion of set A that is not part of set B. Then the rank operator $\mathrm{R}^{(t)}(a)$ assigns to each individual $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ the index i of the corresponding Pareto front $\mathrm{ndom}(M_i)$ 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 (more precisely, the sum of its edge lengths) 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. 5. Then all individuals $a \in \mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$ are sorted in ascending order according to the partial ordering relation $\leq_n$ defined by

$$
\begin{aligned}
a_i \leq_n a_j \Leftrightarrow & \Big(\mathrm{R}^{(t)}(a_i) < \mathrm{R}^{(t)}(a_j)\Big) \text{ or} \\
& \Big(\mathrm{R}^{(t)}(a_i) = \mathrm{R}^{(t)}(a_j) \,\wedge\, C(a_i) \geq C(a_j)\Big) \qquad (3)
\end{aligned}
$$

and the first $|\mathcal{P}|$ individuals form the new parent population $\mathcal{P}^{(t+1)}$. We refer to the described selection method as *NSGA-II deterministic selection* DM throughout this article.

TM—Vector-valued objectives are subject to tournament selection.

The selection method for the second test scenario $\boxed{2}$ is almost completely in accordance with the *NSGA-II deterministic selection*. But it is chosen to be an EP-style tournament selection with 5 opponents to determine the parents $\mathcal{P}^{(t+1)}$ for the next generation from $\mathcal{P}^{(t)} \cup \mathcal{O}^{(t)}$, cf. selection method LA. 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 the partial relation $\leq_n$ of eq. (3). We refer to the described selection method as *NSGA-II tournament selection* TM throughout this article.

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 [14, 28, 42, 30]. Not all operators might be necessary at all stages of evolution. In our 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 presented in [28, 30, 47], which is inspired by Davis' work [10]. The underlying assumption is that recent beneficial modifications are likely to be beneficial in the following generations.

The basic operators that are actually employed in a given optimization scenario are divided into groups, those adding connections, deleting connections, adding nodes, deleting nodes, and solely modifying weights. Let Ω be the set of variation operators, G the number of operator groups used in a task and $p_o^{(t)}$ 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 0.2 in case of the test scenario $\boxed{1}$ where $G = 5$. In case of the face task within test scenario $\boxed{2}$, where no node deletion operators are used ($G = 4$), the probabilities add up to 0.25. In the car task within test scenario $\boxed{2}$ the initial probabilities for *add-connection-P* and *add-node* are set to 0.3, and the initial probability for *delete-connection-P* it is set to 0.4. This produces a slight bias towards deleting connections.

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 we set $\tau = 4$. 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. This is for the scalar value based selection scheme, case LA,

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

and for the vector-valued selection schemes DM and TM,

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

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

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

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 our 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 in comparison 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 [28, 47, 30].

4 Evaluating multi-objective optimization

The performance assessment of stochastic multi-objective algorithms is in general more difficult than evaluating single-objective algorithms. The reason is that in empirical investigations, sets of sets (i.e., the non-dominated solutions evolved in multiple trials of different algorithms) have to be compared. Many ways of measuring the performance of multi-objective optimization algorithms have been proposed, here we use two unary quality indicators, the hypervolume indicator and the additive ϵ-indicator. We concisely define the performance measures used, for a detailed description of the methods we refer to the literature [33, 56, 18].

Given two sets of objective vectors $A, B \subseteq \mathbb{R}^n$ there is a common sense definition of one set being better than the other. Set A is better than B and we write $A \rhd B$ if for every element $\boldsymbol{z} \in B$ there exists an element $\boldsymbol{z}' \in A$ that is not worse than $\boldsymbol{z}$ in each objective, $\forall j \in \{1, \ldots, n\} : z'_j \leq z_j$, and $\operatorname{ndom}(A) \neq \operatorname{ndom}(B)$. Otherwise we have $A \not\rhd B$. However, in general neither $A \rhd B$ nor $B \rhd A$ holds for sets A and B. Therefore, quality indicators are introduced.

An unary quality indicator assigns a real valued performance index to a set of solutions. A single indicator captures only certain aspects (preferences of a decision maker) of the evolved solutions, therefore a set of different indicators is used for the comparison of multi-objective optimization algorithms. Here, the hypervolume indicator [55] and the ϵ-indicator [56] are chosen. We use the performance assessment tools that are part of the PISA [4] software package with standard parameters.

Before the performance indicators are computed, the data are normalized. Assume we want to compare k algorithms on a particular optimization problem. For each algorithm, we have conducted T trials. For comparison, we consider the solutions in the populations after a predefined number g of evaluations. First of all, we consider the non-dominated individuals of the union of all kT populations after g evaluations. Their objective vectors are normalized such that for every objective the smallest and largest objective function value are mapped to 1 and 2, respectively, by an affine transformation. These objective vectors make up the reference set A_{ref}. The mapping to $[1, 2]^n$ is fixed and applied to all objective vectors under consideration.

The hypervolume measure or $\mathcal{S}$-metric was introduced in [55] in the domain of EMO. With respect to the reference point $\boldsymbol{z}_{\text{nadir}}$, it can be defined as the Lebesgue measure Λ of the union of hypercubes in objective space [9]:

$$\mathcal{S}_{\boldsymbol{z}_{\text{nadir}}}(A') = \Lambda\left(\bigcup_{\boldsymbol{z} \in \operatorname{ndom}(A')} \{\boldsymbol{z}' \in \mathbb{R}^n \mid \boldsymbol{z} \prec \boldsymbol{z}' \prec \boldsymbol{z}_{\text{nadir}}\} \right) \ .$$

The hypervolume indicator with respect to reference set A_{ref} is defined as

$$\mathcal{I}_{\mathcal{S}, A_{\text{ref}}}(A) = S_{\boldsymbol{z}_{\text{nadir}}}(A_{\text{ref}}) - S_{\boldsymbol{z}_{\text{nadir}}}(A) \ .$$

The reference point $\boldsymbol{z}_{\text{nadir}}$ is an objective vector that is worse in each objective than all individuals (here $\boldsymbol{z}_{\text{nadir}} = (2.1, \ldots, 2.1)$). A smaller value of $\mathcal{I}_{\mathcal{S}}$ is preferable. The additive unary ϵ-indicator is defined as

$$\mathcal{I}_{\epsilon, A_{\text{ref}}}(A) = \inf \{\epsilon \in \mathbb{R} \mid \forall \boldsymbol{z} \in A_{\text{ref}} \, \exists \boldsymbol{z}' \in A \, \forall i \in \{1, \ldots, n\} : z_i \geq z'_i - \epsilon\} \ .$$

Basically, the ϵ-indicator determines the smallest offset that has to be subtracted from the fitness values of the elements in A such that the resulting Pareto front covers the Pareto front of A_{ref} in objective space. A smaller value of $\mathcal{I}_{\epsilon, A_{\text{ref}}}$ is preferable.

In [33] and [56], various properties of quality indicators are studied. Of particular interest is the relation to the “being better” definition given above.

An unary quality indicator is $\ntriangleright$-compatible if a better indicator value for A than for B implies $B \ntriangleright A$. An indicator is $\triangleright$-complete, if $A \triangleright B$ implies a better indicator value for A than for B. Both the ϵ-indicator as well as the hypervolume indicator are $\ntriangleright$-compatible and $\triangleright$-complete.

5 Experimental setup

In all experiments, the FFNNs have at most one hidden layer and the activation functions are of logistic sigmoidal type. In all cases we simulate $T = 10$ trials. For each trial we set $|\mathcal{P}^{(t=0)}| = 25$.

In the car task, each FFNN 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. We refer to this architecture as the *car reference topology*, see section 3.1. In the face tasks, $\mathcal{P}^{(t=0)}$ consists of copies of the 400-52-1 architecture of [40], the *face reference topology*, each of which is randomly initialized like the *car reference topology* at $t = 0$.

Although cross-validation is applied when training the FFNNs, the evolutionary (LA, DM, TM) or the pruning optimization (PR) may lead to overfitting, in our case they may overfit to the patterns of $D_{\text{train}} \cup D_{\text{val}}$. Hence, we additionally introduce the data set D_{test} to finally choose models that generalize well and store those in the external archive $\mathcal{A}^{(t)}$ if their objective vector $\boldsymbol{z}$ computed from D_{test} is not dominated by any member of $\mathcal{A}^{(t)}$. Members that are dominated by $\boldsymbol{z}$ are removed from the archive. That is, we use D_{test} for some kind of cross-validation of the evolutionary or pruning process. 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 $\mathcal{A}^{(t_{\max})}$.

We train 2000 instances of the *face reference topology* and the *car reference topology* for 100 training steps using the improved Rprop learning procedure on D_{train}. From all instances and all training steps we select the networks a_{ref} (for the face and the car tasks, respectively) with the smallest classification error on $D_{\text{val}} \cup D_{\text{test}}$. When selecting the *reference topologies* a_{ref}, we decide in a similar way as in picking a solution from the evolved or pruned architectures, but taking also D_{val} into account. This is reasonable, since D_{val} has not been applied during FFNN training.

In the following figures 6 and 7, results are normalized by the performance of the *face reference topology* and *car reference topology* a_{ref}, respectively.[4] For example, the normalized classification error of an FFNN a is given by $\text{CE}'_a(D) = \text{CE}_a(D)/\text{CE}_{a_{\text{ref}}}(D)$ and the normalized number

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

of connections and number of nodes by $n'_{\mathrm{con}}(a) = n_{\mathrm{con}}(a)/n_{\mathrm{con}}(a_{\mathrm{ref}})$ and $n'_{\mathrm{hid}}(a) = n_{\mathrm{hid}}(a)/n_{\mathrm{hid}}(a_{\mathrm{ref}})$, respectively.

5.1 Test scenario $\boxed{1}$

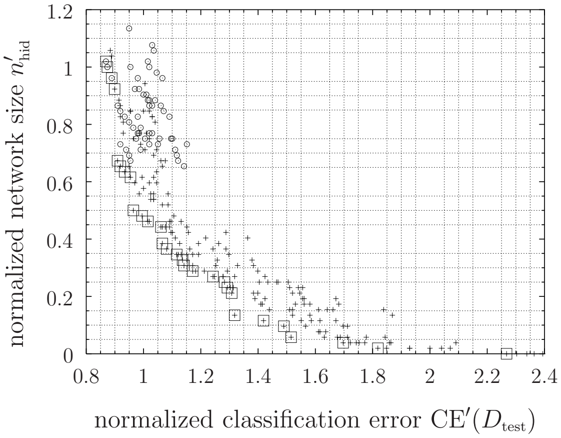
This scenario is concerned with the face task only. It has already been shown that the size of the face detection network of the Viisage-FaceFINDER® system can be successfully reduced without loss of accuracy by the scalar fitness value approach LA [48]. Here, we investigate whether we can improve these results by using the vector valued selection DM, see [47]. This is done by comparing the performance of our hybrid algorithm using either selection variant LA or DM. We strongly expect a result in favor of method DM since a single-objective method (LA) is evaluated by multi-objective performance measures which are made necessary by the nature of the problem. At the time this analysis was conducted, our goal was to challenge the popular single-objective approach to FFNN structure optimization.

We assume that the runtime of our algorithm is strongly dominated by the number of fitness evaluations (due to the efforts spent for learning) and that the number of allowed fitness evaluation is fixed. Then there is roughly no difference in runtime between the single-objective (LA) and the multi-objective (DM) approach.

All T trials of both variants LA and DM of the evolutionary algorithm described above are run for $t_{\max} = 200$ generations (i.e., 5025 fitness evaluations per trial) using only the face task. For both methods LA and DM we use the objective vector $\boldsymbol{z}_a = (n_{\mathrm{hid}}, \mathrm{CE}(D_{\mathrm{test}}))$ of every individual $a \in \mathcal{P}^{(t)}$ to iteratively update the external archive.

5.2 Test scenario $\boxed{2}$

In this scenario we aim to show that the significantly higher effort of implementing and applying advanced EMO by method TM can be worthwhile compared to the simpler pruning method PR [22]. Both the car and the face task are considered for this investigation: methods are compared within tasks, and the results of the within-task comparisons are contrasted. All trials T of method PR are performed for $t_{\max} = 90$ iterations at $p = 10\%$.[5] All TM trials are performed for $t_{\max} = 200$ generations. For both methods PR and TM we compute the objective vector $\boldsymbol{z}_a = (n_{\mathrm{con}}, \mathrm{CE}(D_{\mathrm{test}}))$ of every individual $a \in \mathcal{P}^{(t)}$ to update the external archive.



indicator	method	value
$\mathcal{I}_\epsilon$	LA	0.745
$\mathcal{I}_\epsilon$	DM	0.138
$\mathcal{I}_\mathcal{S}$	LA	0.673
$\mathcal{I}_\mathcal{S}$	DM	0.137

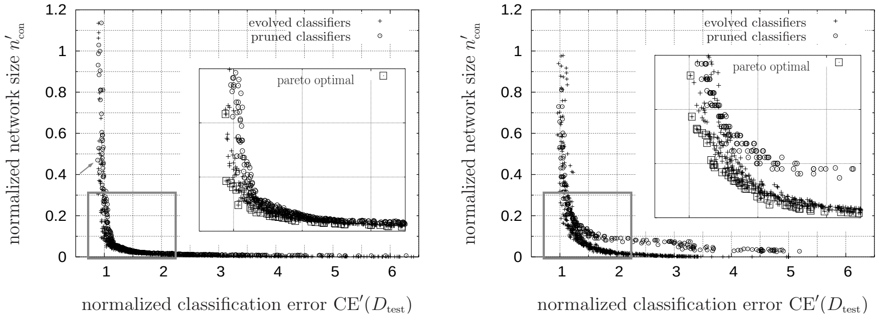
Fig. 6. Evolved solutions by selection variants LA and DM in test scenario $\boxed{1}$. The left plot shows the two objectives, the normalized classification error $\mathrm{CE}'(D_{\mathrm{test}})$ and the normalized number of hidden neurons n'_{hid}, for all FFNNs of all Pareto fronts of all trials. The circles represent the outcomes of selection method LA, and the crosses the results of the variant DM. The non-dominated FFNNs from the union of all trial outcomes from both methods (the "meta Pareto front") are highlighted. The right table shows the results of performance indicators explained in the text. The distributions over all trials of indicator results for methods LA and DM differ in a statistically significant way when compared by the Wilcoxon rank sum test ($p < 0.005$).

6 Results

The normalized results of test scenario $\boxed{1}$ are shown in Fig. 6. Indicator results confirm what is evident from visual inspection of the two Pareto fronts: method DM performs considerably better than method LA.

The normalized results of test scenario $\boxed{2}$ are shown in Fig. 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 and the errors of the generated FFNNs 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, EMO is clearly superior. In both tasks, the distributions of the ϵ- and hypervolume indicators differ in a statistically significant way. All results persist when considering the vectors $(n_{\mathrm{hid}}, \mathrm{CE}(D_{\mathrm{ext}}))$ and $(n_{\mathrm{con}}, \mathrm{CE}(D_{\mathrm{ext}}))$ instead of $(n_{\mathrm{hid}}, \mathrm{CE}(D_{\mathrm{test}}))$ and $(n_{\mathrm{con}}, \mathrm{CE}(D_{\mathrm{test}}))$, respectively. This shows that no significant overfitting has occurred.

[5] No regular FFNNs were ever produced afterwards. Reducing the pruning percentage p to the point where valid FFNNs could be produced for 200 generations did not change results.



indicator	method	value
$\mathcal{I}_\epsilon$	PR	0.0293
$\mathcal{I}_\epsilon$	TM	0.0190
$\mathcal{I}_\mathcal{S}$	PR	0.0277
$\mathcal{I}_\mathcal{S}$	TM	0.0122

indicator	method	value
$\mathcal{I}_\epsilon$	PR	0.0842
$\mathcal{I}_\epsilon$	TM	0.0708
$\mathcal{I}_\mathcal{S}$	PR	0.1118
$\mathcal{I}_\mathcal{S}$	TM	0.0606

Fig. 7. Evolved solutions by selection variants TM and PR in the second scenario $\boxed{2}$. Left: results from the car task. Right: results from the face task. Shown on top are the unions of all trial outcomes; members of the meta Pareto fronts are shown in the magnifications. The only pruned FFNN in the meta Pareto front of the car task is indicated by an arrow. The performance indicators (tables at the bottom) are explained in section 4. The distributions over all trials of indicator results for methods PR and TM differ in a statistically significant way when compared by the Wilcoxon rank sum test ($p < 0.005$).

7 Discussion

When considering scenario $\boxed{1}$, it is evident that method DM performs better, but this is not surprising since we evaluated a single-objective method in a multi-objective setting. Because a prior decision about the relative weighting of objectives was taken before optimization when using method LA, it can be expected to give adequate results in the corresponding region of objective space, whereas solutions in other regions are less likely to be selected. In contrast, methods like DM and TM can select solutions from all regions of objective space: when using multi-objective performance indicators, which measure performance over all of objective space, this leads to the significant differences in performance that are observed. Single-objective methods like LA are useful whenever the desired trade-off between objectives is known a priori. However, this will not usually be the case, and therefore multi-objective optimization is clearly the strategy of choice.

Considering test scenario $\boxed{2}$, we interpret the result of the car task as an indication that the problem class is intrinsically easier (w.r.t. the magnitude of the classification error of the best conceivable FFNN) than the face task since both the ϵ- and the hypervolume indicator results of both methods lie more closely together. This suggests similar performance.[6] We assert that the simpler optimization method can yield competitive performance on simpler tasks. For the more difficult face task, a sophisticated (here: evolutionary) optimization strategy is clearly favorable.

For the support of our interpretation about the difficulty of both tasks in scenario $\boxed{2}$, we observe that the (absolute) error $\mathrm{CE}(D_{\mathrm{test}})$ of the best trained car reference topology is 3.5 times smaller than $\mathrm{CE}(D_{\mathrm{test}})$ of the corresponding face reference topology (see section 5). 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 performance should be considered meaningful. Furthermore, optimization in the car task produced FFNNs without a hidden layer, which nevertheless had an (absolute) classification accuracy of about 80%. We take this as a hint that the problem is almost linearly separable and therefore can be considered “easy”.

Finally, we want to compare the performance of the methods PR and LA. Both are essentially single-objective methods, since they compute a scalar fitness value from a weighted average of objectives. In the case of LA, this is trivially fulfilled by construction, see eq. (2), whereas pruning takes the coefficient of one objective (classification error) to be zero and focuses on the other one (number of connections) only. However, pruning does not perform selection at all since it simply copies the current population into the next. Thus, the success of structure adaptation is not taken into account. Therefore, there is no bias towards certain trade-off solutions in selection. In contrast, method LA uses a scalar fitness value to select the individuals of the next parental population. Hence, only individuals optimized for a single fixed trade-off between objectives are archived. The comparison between these single-objective methods is possible by contrasting them to their advanced multi-objective counterparts TM and DM and comparing the results. It turns out that the performance differences between methods LA and DM are much more pronounced considering any of the given performance indicators than the corresponding differences between methods PR and TM. This supports our previous explanations and shows that pruning, although an amazingly primitive method by itself, can significantly profit from multi-objective performance evaluation. To conclude, when using optimization methods in a multi-objective setup, great care must be taken not to introduce a bias restricting search to certain regions of objective space.

[6] The absolute values of the performance indicators depend on reference sets. When comparing two performance indicator values for one task, one should therefore look at their difference, which is independent of the reference set, and not at their quotient.

Acknowledgment

Christian Igel acknowledges support from Viisage Technology AG under contract “SecureFaceCheck”.

References

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. S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa – A platform and programming language independent interface for search algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 494 – 508. Springer-Verlag, 2003.
5. H. Braun. *Neurale Netze: Optimierung durch Lernen und Evolution*. Springer-Verlag, 1997.
6. 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, 2003.
7. 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.
8. 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.
9. C. Coello Coello, D. Van Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-objective Problems*. Kluwer Academic Publishers, New York, 2002.
10. 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.
11. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.
12. 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.
13. 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.

14. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
15. D. Ferster and K. Miller. Neural mechanisms of orientation selectivity in the visual cortex. *Annual Review of Neuroscience*, 23:441–471, 2000.
16. J. E. Fieldsend and S. Singh. Pareto evolutionary neural networks. *IEEE Transactions on Neural Networks*, 16(2):338–354, 2005.
17. D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.
18. C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Presented at the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), 2005.
19. 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.
20. 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.
21. 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.
22. A. Gepperth and S. Roth. Applications of multi-objective structure optimization. In M. Verleysen, editor, *5th European Symposium on Artificial Neural Networks (ESANN 2005)*, pages 279–284. d-side Publications, 2005.
23. 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.
24. E. Hjelmas and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.
25. H. M. Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, Germany, 1994.
26. 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.
27. C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.
28. 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.
29. C. Igel and B. Sendhoff. Synergies between evolutionary and neural computation. In M. Verleysen, editor, *13th European Symposium on Artificial Neural Networks (ESANN 2005)*, pages 241–252. d-side Publications, 2005.
30. 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.

31. 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 2004)*, pages 1–8. IEEE Press, 2004.
32. J. Kaszubiak, M. Tornow, R. Kuhn, B. Michaelis, and C. Knoeppel. Real-time vehicle and lane detection with embedded hardware. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 619–624, 2005.
33. J. D. Knowles and D. W. Corne. On metrics for comparing non-dominated sets. In *Congress on Evolutionary Computation Conference (CEC 2002)*, pages 711–716. IEEE Press, 2002.
34. 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.
35. 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.
36. 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.
37. L. Prechelt. Early stopping – but when? In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *LNCS*, chapter 2, pages 57–69. Springer-Verlag, 1999.
38. R. D. Reed and R. J. Marks II. *Neural Smithing*. MIT Press, 1999.
39. M. Riedmiller. Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(5):265–278, 1994.
40. 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.
41. 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.
42. J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.
43. 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.
44. 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.
45. Viisage Technology AG. `http://www.viisage.com`.
46. 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.
47. 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.

48. 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.
49. 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.
50. J. M. Wolfe. Visual search. In H. D. Pashler, editor, *Attention*. London UK: University College London Press, 1998.
51. 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.
52. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
53. G. P. Zhang. Neural Networks for Classification: A Survey. *IEEE Transactions on System, Man, and Cybernetics – Part C*, 30(4):451 – 462, 2000.
54. W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399 – 458, 2003.
55. E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, pages 292–301. Springer-Verlag, 1998.
56. 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.