IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

Pruning of Neural Networks

G. Thimm and E. Fiesler

IDIAP–RR 97-03

February 1997

Dalle Molle Institute for Perceptive Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

IDIAP Research Report 97-03

Pruning of Neural Networks

G. Thimm and E. Fiesler

February 1997

hoosing a suitable topology for a neural network, given an application, is a difficult problem. Usually, after a tedious trial-and-error process, an oversized topology is chosen, which is prone to various drawbacks like a high demand on computational resources and a high generalization error. A way to solve this is to trim the network size during the training process. This is done with so-called *pruning* methods, of which an overview is given.

From these methods, those that are potentially suitable for high order perceptrons are selected, and then adapted accordingly. Next, they are tested on a variety of benchmarks by means of a large number of experiments. The conclusions are both of a generic nature, pointing out some pitfalls of neural network pruning in general, and of a more specific nature, identifying the best pruning methods for high order perceptrons.

Keyworks: optimal network size, generalization performance, pruning, network topology, neural network optimization, high order perceptron.

1 Introduction

One of the most important problems encountered in the practical application of neural networks is to find a suitable or, ideally, minimal neural network topology. Some of the main reasons are that an unsuitable topology increases the training time or even causes non-convergence, and that it usually decreases the generalization capability of a network [1]. Additionally, there are economical and technical arguments to prefer small networks: the price for hardware implementations is directly related to the surface or number of chips and therefore the network size. Similarly, software implementations for oversized neural networks are far less efficient.

A basic approach to find nearly minimal topologies are, apart from *constructive* or *growing approaches* [2], *pruning* methods. Generally, the latter method starts the training with a neural network which is expected to be big enough to ensure a successful training. Then, when the neural network is estimated being too big or simply when the network training is successful, some connections or neurons are removed and the training resumed. If the retraining converged (the convergence criterion is fulfilled before a predefined number of training cycles passed), the removal-retraining cycle is resumed. If the retraining failed, the smallest network, that performed according to the given convergence criterion, is assumed to have the most suitable topology for the given data set.

Independent of the pruning method, the evaluation of the quality of the networks it produces is difficult to perform, mainly because of a lack of neural network *optimality criteria*. A primary criterion, the *minimal network topology*, can be defined but is very complex to determine for a specific application [3]. Obtaining a proof for a certain neural network topology being actually minimal, is usually infeasible. Besides this, the minimality criterion varies for different implementations and applications. Such a criterion can be based on the number of layers, neurons, and connections, or a mixture of these. Furthermore, various implementation dependent constraints may play a role in the choice of topology: for some types of hardware implementations not only the total number of connections and neurons is important but also, for example, the maximal number of connections going to and from a neuron (these numbers are often called the *fan-in* and the *fan-out* of a neuron). Or, if the application has tight real-time requirements, the depth of a multilayer perceptron may be bound.

Besides the topology, the other important criterion, *generalization performance*, is, apart from being improperly defined, also application dependent. In general, tradeoffs between training time, network size, and generalization performance should be taken into account when comparing pruning methods, which adds another dimension to the problem of defining an optimal neural network.

Numerous pruning (and growing) algorithms have been proposed, but their efficiency is usually not compared with each other. Reviews of pruning methods and heuristics, without experimental evaluation, are for example given in [4] and [5]. It is infeasible to compare methods for the optimization of neural networks in a theoretical way. The comparison of these methods is therefore usually performed empirically. This requires optimality criteria to be established, plus a framework defining the details of experiments in which the pruning methods are applied to neural networks. This framework should include factors like the generalization performance of the network, the total training time, the complexity of the pruning method, and the implementation costs. Furthermore, results need to be supported by the analysis of a large number of experiments, guaranteeing a high statistical confidence level. This, however, is often neglected [6]. A framework, that can be used for the comparison of pruning methods, is presented in the following and applied to high order perceptrons. This or a similar framework is applicable to other types of neural networks and constructive methods.

2 Connection Pruning Methods

The main ingredients of a pruning algorithm concerns the decisions: *which unit(s)* to prune, *when to prune*, and *when to stop* the training. The choice of the method, which selects the units to be removed, is the most crucial part. Since optimal choices for when to prune and to stop it are unlikely to be greatly influenced by the choice of the pruning method, this research is limited to the first question. A simple strategy is used for the second and third decision: a network is pruned by one or a few connections, whenever its training converged. Then, the training is eventually aborted when the network ceases to learn and the network with the best performance (in whatever sense) so far is kept and recalled at the end of the training. This approach has another advantage: as the networks are pruned until they cease to re-learn the training data, it is possible to verify whether some of the assumptions made for other types of neural networks also apply to high order perceptrons (see section 4).

For comparison, five weight removal heuristics were chosen based on their low computational complexity and applicability to high order perceptrons (excluding second order methods). These methods emerge from the idea to minimize the error induced by the removal of the unit or, in other words, to estimate the *sensitivity* of the neural network to the removal of a certain connection.

1. The simplest heuristic selects the connections with the smallest weights. In addition to commonly used method, which removes only connections, the growth of the error is reduced by adding the connection's *mean contribution* $\frac{1}{P}\sum_{p=1}^{P} c^p$ (its output c averaged over the training set of size P) to the bias of the neuron receiving input from the removed connection.

2. J. Sietsma and R. J. F. Dow proposed to remove the connections with the smallest contribution variance $\sigma_p(c^p)$ [7]. The method is therefore called the smallest variance ($min(\sigma)$) method. The mean output of the removed connection is added to the corresponding bias.

3. E. D. Karnin estimates the sensitivity s of a connection by:

$$s = \sum_{n=1}^{N} (\Delta w(n))^2 \frac{w(n)}{\eta(w(n) - w(0))},$$

   where $w(n)$ is the weight in the current training epoch n, $w(0)$ the initial weight, and $\Delta w(n)$ the weight change in the n-th epoch [8]. This formula has a problem: the denominator can become zero and experiments show that this sometimes happens. As E. D. Karnin neglects this case in his publication, it is decided here to set in this case the whole fraction to zero. s is therefore calculated using:

$$s = \sum_{n=1}^{N} \begin{cases} (\Delta w(n))^2 \frac{w(n)}{\eta(w(n)-w(0))} & \text{if } w(n) \neq w(0) \\ 0 & \text{otherwise.} \end{cases}$$

4. M. C. Mozer and P. Smolensky developed a weight removal method called *skeletonization* which estimates the error induced by the removal of a unit by multiplying its output with an *additional strength*. Then, with having in mind that setting an additional strength of zero is equivalent to removing it, those units are removed for which the derivative of the error function to these

additional strengths are small (actually, they use an exponentially decaying average of these values) [9].

5. W. Finnoff *et al.* define a test statistic based on the probability that a weight becomes zero. A connection is removed if the probability that it becomes zero is high. This sensitivity measure is integrated into a pruning method called *autoprune* [10]. This pruning method is extended to the pruning method $\lambda-prune$ by L. Prechelt in order to determine how many units should be pruned at each step [11]. However, in the experiments described in the following, only the test statistic is used as sensitivity measure.

The reason for not using the whole framework of a pruning method is the difficulty to find good settings for the various additional parameters involved in the algorithms which decides upon when to prune and when to stop the training. This problem became apparent during a project on pruning multilayer perceptrons [12]. These parameters also strongly affect the user-friendliness and inhibit a fair comparison with the embedded sensitivity estimation. Note that the framework of E. D. Karnin, which J.-L. Beuchat found to be best for multilayer perceptrons, is incompatible with the theoretical justification of OBD, as the network is pruned before it converges: OBD assumes that $\Delta\mathbf{w} = 0$, or equivalent, that the network has converged into a (local) minimum.

Other pruning methods were excluded from this study because they are either unsuitable for high order perceptrons or have a high computational complexity. The latter class includes methods using a full Hessian matrix, like for example flat minimum search [13] and Optimal Brain Surgeon (OBS) [14].

Weight decay was excluded, as it was found to be not effective by S. J. Hanson *et al.* [15]. Their publication also throws a negative light on methods using penalty terms[1]. They observed that about 75% of the simulations where weight decay was used failed to converge, whereas backpropagation without a penalty term always converged under the same conditions. Similar methods are summarized by E. D. Karnin [8] and G. Fahner [16]. Why methods using penalty terms may fail to converge is exemplarily shown for the penalty term proposed by A. S. Weigend *et al.* [17]:

$$\lambda \sum_i \frac{\frac{w_i^2}{w_0^2}}{1 + \frac{w_i^2}{w_0^2}}$$

In order to permit an evaluation of this formula, let the value of w_0 in this penalty term be 1 and the weight of the penalty term λ in the objective function be 0.4. Then the penalty term has its minimum at $\mathbf{w}=0$ and approaches $\lambda=0.4$ for large weight vectors (compare the dashed curve in figure 1). It is obvious that if this term is included into the objective function, then minima of the error term, which are equivalent to a weight vector close to $w=0$, are preferred. Now, suppose that the error surface is shaped as the solid curve in figure 1 with a global minimum for $w \approx 2$. The addition of these two curves illustrates the impact on the (fictive) error surface represented as the dotted curve: the former harmless local minimum at $w \approx 0.3$ replaces the former global minimum, the network will therefore very often fail to learn properly.

3 Experiments

The major aim of the experiments described in the following sections is the determination of the best pruning method(s) with in respect to the network size and generalization performance. Furthermore, some side-issues are taken into consideration:

- Is the distribution of the network sizes obtained by a certain pruning method comparable to another? I.e., if the average network size obtained with method A is smaller than for method

[1] Penalty methods are based on modifications of the objective function designed to penalize certain weights.

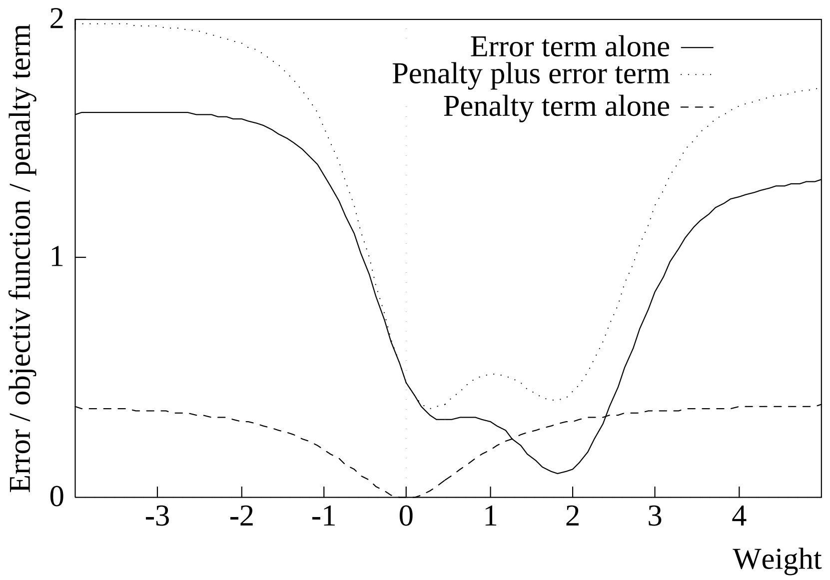

Figure 1: The penalty term may change the location of global minima.

B, can it be expected that the smallest network obtained during a series of simulations obtained with method A is also smaller than for method B?

- Are networks with the smallest size those with the best generalization performance?
- Does the generalization performance change in function of the network size?
- Has the initial size an influence on the final networks?

The experiments analysed in the following consist of at least 100 simulations each; if the outcomes of two experiments are compared, the confidence in the difference of their respective results was calculated and ensured to be at least 99%.

Data set	Precision on training set
Solar	MSE smaller than 0.05
CES	MSE smaller than 0.08
Monk 1-3	100% correctly classified
Auto-mpg	MSE smaller than 0.06
Glass	MSE smaller than 0.03
Servo	MSE smaller than 0.05
Wine	100% correctly classified
Digits	99% correctly classified

Table 1: The required precision on the training sets for the pruning experiments.

Each simulation performed follows the same scheme:

1. Initialization of the network of order ω with random weights (compare [18]). During a real application of high order perceptrons, the order ω would be initially chosen to be one and then increased by one whenever it is clear that the corresponding network is unable to learn the task. However, for research purposes, various orders are examined (see for example table 2).

2. Application of the backpropagation algorithm until the in table 1 given percentage of training data is correctly classified by the network, respectively the mean square error on the training set reaches the value listed in this table (see table 2 for the learning rate used).

3. Removal of connections from the network. Depending on the size of the network, from one connection up to 5% of the connections are removed from the network (see Remark 2 below). The connection(s) to be removed are those found to have the smallest sensitivity calculated according to the methods listed in section 2.

4. If required, the network is retrained until it reached the criterion as listed in table 1, or a certain limit of training steps has passed[2]. If the network converged, pruning is resumed (step 3). If not, the last network found is accepted as a solution.

Remark 1: for data sets with a high number of zero elements per input vector, as for example the Digits data set, many second order connections have a constant value of zero for all inputs and can therefore be removed without loss. However, this was not done in these experiments in order to test whether the pruning methods are able to remove those connections without further intervention.

Remark 2: if the pruned network is very large as compared to the minimal network, the pruning of a connection often did not require any retraining. In the simulations described in the remainder of this chapter, the error remains usually almost unchanged during the initial phase of a simulation and therefore did not trigger a retraining of the network.

In order to accelerate the simulations, a small percentage (1%–5%) of the connections was therefore removed, as especially the bigger neural networks require a considerable simulation time.

Remark 3: as the primary aim of this research is to find networks of minimal size, no pruning was done before the network had converged. Other researchers regard pruning also as a possibility to speed up learning and remove units before the network converged [10] [11], taking the risk to find neural networks of a sub-optimal size or generalization performance.

3.1 Minimizing Network Size

As the primary aim of pruning is the construction of small networks, pruning methods are usually compared by means of the average final network size.

But this measure neglects that in real applications the training time may or may not be important. Examples are applications for which the final network size is crucial, and the total training time allows the performance of several training sessions. Then, instead of the average network size, the percentage of networks close to an optimal size is important, as a high average can also indicate that some training sessions ended up with oversized networks. The other extreme is that the training of a neural network is very time consuming and can not be repeated often. In this case, rather than the average network size of the percentage of almost optimal neural networks, the amount of largely oversized networks is of interest.

An example justifying this consideration is the outcome of the experiment with a 4th order network applied on the CES data set (see figure 2; the horizontal axis shows the number of connections in the final networks and the vertical axis the percentage of simulations with final networks of this size). Both the smallest weight removal method and the smallest contribution variance method result in networks of an average size of 7.6 connections. Whilst the smallest weight removal method reached in more than 10% of the simulations a final network size of 4 connections, the smallest contribution variance method never produced a network with less then 5 connections (an experiment consists of 200 simulations). A second example is an experiment performed with the Monk 2 data set and a second order perceptron, see figure 3: the best result for the smallest weight removal method is a network with 40 connections (in 21% of 100 simulations), and 30 (in 23% of 100 simulations) for the $min(\sigma)$ method.

The following performance measure takes these observations into account:

[2] This limit was conservatively chosen, to ensure that the minimal network found could not be pruned any further.

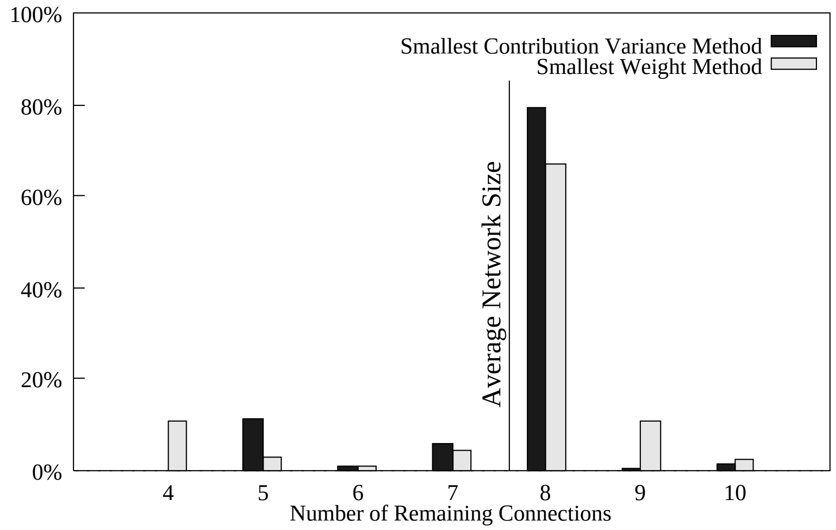

Figure 2: The distribution of the final network sizes for the CES data set.

Definition: (10% and 90% limit for network size)

The 10% limit, respectively the 90% limit, is an upper limit for the network size which is reached in at least 10%, respectively in at least 90%, of the simulations per experiment.

The network size in the definition can be, for example, the final network size or the size of the network with the best generalization performance.

Example: in figure 2 it can be seen that about 10% of the final networks pruned with the $min(\mathbf{w})$ method have 4 connections, and the 10% limit is therefore 4. As the $min(\sigma)$ method never produces networks of 4 connections but a reasonable amount of networks of size 5, the 10% limit is 5. Similarly, the 90% limits differ (they are 9, respectively 8).

Using this measure, method A is judged better than method B, if method A has a smaller 10% (90%) limit, or the methods have the same 10% (90%) limit, and method B produces less networks than method A with this maximal size (assumed that the confidence in this difference is at least 99%; compare appendix A).

Depending on whether the 10% or the 90% limit is applied, either the $min(w)$ or the $min(\sigma)$ pruning method appears to be more effective in the example described above. Usually this discrepancy is observable if the average performance of the two methods is similar.

In table 2, five methods are compared[3] using the 10% and 90% limit criteria on several data sets and networks of different order (Ω is the order of the initial network; 2r means that the network is of order two, but not fully connected[4]). The best result for a certain combination of data set and network order for the 10% and 90% limit is marked in bold type face. Also marked in bold font are results for which the confidence that they are worse than the best result within the same series of experiments could not be rejected with a confidence of at least 95% (compare appendix A). The exclamation marks show the lines for which the best method using the 10% and the 90% limit measure results in a different judgement.

The table shows clearly, that the method of E. D. Karnin and the *skeletonization* method are surprisingly worse than the smallest weight and smallest contribution variance method. Whereas the

[3]Not all networks were pruned using all sensitivity estimators as it became clear during the experiments that these methods (see table 2) do not perform well, and the available computation resources were limited.

[4]In order to reduce the network to a size that permits the performance of the experiments, only those second order connections are used which take both inputs in the same row or the same column of the image. The performance of the resulting network remains considerably high.

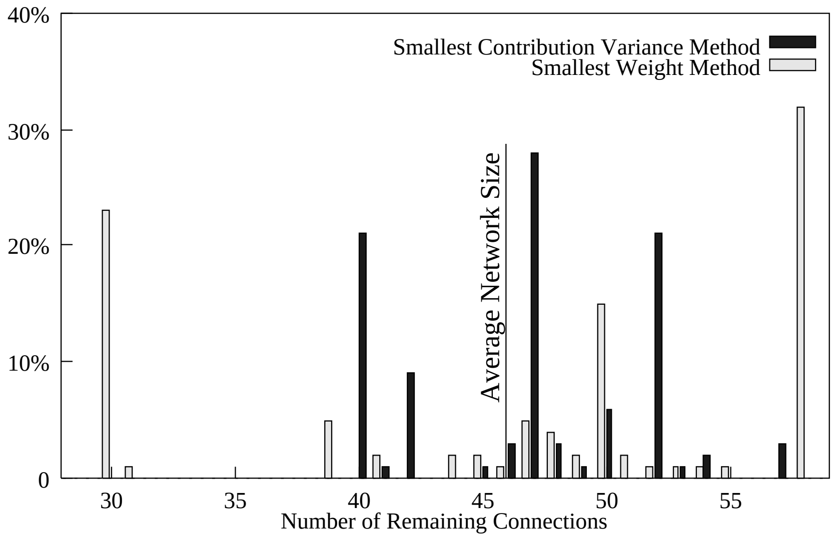

Figure 3: The distribution of the final network sizes for the Monk 2 data set.

autoprune method of W. Finnoff *et al.* and the smallest weight removal methods perform almost equally in all the cases.

3.2 Generalization Performance

The phenomenon of over-fitting in neural networks with a static topology is usually prevented by *early stopping* [17]. Although this approach seems to be applicable in combination with neural network pruning, it is shown not to have the desired effect: even though the network pruning can improve the generalization performance, the retraining usually degrades it. This means that two opposite forces influence the neural network behavior. Furthermore, pruning is not formally shown to impose smoothness onto the function represented by a neural network, as it has been done for the regularization approach (see [19] for a review of regularization applied to neural networks).

The common belief in the neural network community is that pruning finally will win the race. In order to validate this, suppose that pruning is an efficient regularization technique (in the sense that it imposes smoothness on the function represented by the neural network). Then the following should be observable:

- The error on a generalization test set should decrease or remain constant each time a pruning - training cycle is completed successfully (taking some small fluctuations into account).
- The generalization performance of the smallest network found should usually be the best among all successful retraining steps during one simulation.

An examination of the results of the experiments performed during this study shows that this is not the case. The changes in the generalization performance during the training are significant and usually non-monotonic. Figure 4 shows three curves, each describing a typical development of the generalization performance for simulations using the Glass data set. The vertical axis in figure 4 depicts the error on the generalization test set after each successful retraining step, and the horizontal axis the number of remaining connections. The initial networks are fully connected second order perceptrons with 121 connections. The connections are pruned using the smallest weight method to an average number of 9. The final networks obtained during the three the simulations are marked by a big black dot.

	Ω	η	10% limit for method					90% limit for method					
			min(σ)	*min*($\mathbf{w}$)	Karnin	*skelet.*	*autoprune*	*min*(σ)	*min*($\mathbf{w}$)	Karnin	*skelet.*	*autoprune*	
Solar	2	0.5	**6**	10	18	62	10	**9**	10	24	76	10	
	3	0.5	**6**	8	16	231	8	**10**	11	24	248	11	
	4	0.1	**8**	13	20	—	13	**13**	18	28	—	18	
Wine	2	0.5	19	**18**	53	53	**18**	**24**	**23**	84	134	25	!!
CES	2	0.1	**3**	**3**	4	4	**3**	**4**	**4**	4	5	**4**	
	3	0.1	**5**	6	6	7	6	**6**	**6**	6	8	**6**	!!
	4	0.1	5	**4**	7	10	**5**	**8**	9	10	13	9	!!
Servo	2	0.01	10	**8**	19	86	**8**	10	**9**	27	89	**9**	
	3	0.01	**22**	33	82	—	33	47	**40**	105	—	**40**	!!
Vowels	2	0.5	**52**	59	126	—	—	**69**	78	186	—	—	
Auto-mpg	2	0.03	13	**8**	13	33	**8**	13	**8**	18	35	**8**	
	3	0.03	9	**6**	14	—	**6**	14	**7**	21	—	**7**	
Glass	1	0.005	**5**	6	12	13	6	**5**	6	14	16	6	
	2	0.005	8	**7**	33	52	**7**	11	**11**	42	78	**11**	
	3	0.005	9	**6**	62	—	**6**	15	**11**	86	—	**11**	
Monk 1	2	0.01	**4**	**4**	44	**4**	**4**	**4**	**4**	80	**4**	**4**	
	3	0.01	**4**	**4**	78	—	—	**4**	**4**	137	—	—	
Monk 2	2	0.01	**30**	40	72	122	40	58	**52**	111	128	**52**	!!
	3	0.01	**28**	**27**	113	—	**28**	**35**	**35**	180	—	**35**	
Monk 3	2	0.01	**11**	11	—	—	11	**11**	12	—	—	12	
	3	0.01	**11**	12	67	15	12	18	19	105	26	**17**	!!
Digits	2r	0.005	**119**	**121**	—	—	**121**	**150**	**153**	—	—	**156**	

"—" means that no experiment was performed

Table 2: Sizes of the smallest networks obtained by pruning networks of order ω in the 10% and 90% limit.

It can be seen that, especially when a network approaches its final size, the generalization performance changes unpredictably and drastically. The main tendency is however towards a better performance for two of the examples, namely those depicted by the straight and the dotted curve in figure 4. The dashed curve shows that the training and weight removal may not lead to a good generalization performance for the smallest network, even if intermediate networks perform well. Such cases are marked by the gray background in figure 4. Also, the generalization performance of the trained but unpruned network is not always worse, in a few cases even better than the performance of the smallest network found by the five pruning methods used in this study.

Table 3 shows the percentage of simulations per experiment where the smallest network has a generalization performance which is at least as good as those of bigger networks during the same simulation. From this table it is clear that the observation described in the last paragraph applies to all of the five methods used in this study. A similar observation was made by L. Prechelt for other pruning methods applied on multilayer perceptrons [11]. The common belief that the generalization capability of a neural network increases independently from the pruning method and data set with a decreasing size of the network can therefore be rejected.

Hence, for practical applications which require networks of minimal size and good generalization capabilities, the best intermediate network up to a certain time needs to be stored. However, as the importance assigned to the generalization performance and the network size is highly application dependent, no universal halting criterion can be defined. As it is impossible to perform an evaluation of the experiments with respect to all variations of such criteria, two extreme cases are considered in the

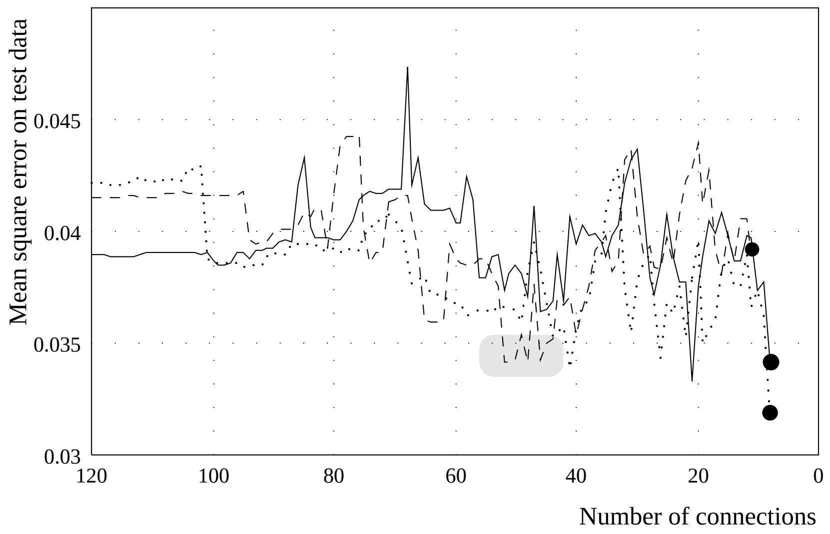

Figure 4: The evolution of the generalization performance during a pruning session.

following sections: the smallest network per simulation and the network with the best generalization performance per simulation.

Furthermore, analogous to the network size as discussed in the previous section, not the average generalization performance might be of interest but the generalization performance obtained in at least 10%, respectively 90%, of the simulations. The corresponding definitions are:

Definition: (10% and 90% limit for generalization)

The 10% limit, respectively the 90% limit, for the generalization performance of a neural network is the upper limit for the generalization performance which is reached in at least 10%, respectively in at least 90%, of the simulations per experiment.

Using this measure, method A is judged better than method B, if method A has a smaller 10% (90%) limit, or the methods have the same 10% (90%) limit, and method B produces less networks than method A with this generalization performance (assumed that the confidence in this difference is at least 99%; compare appendix A).

Table 4 shows the generalization performance for the smallest network per simulation and table 5 the best generalization performance per simulation. From these tables it is clear that, similar to the observations made for the final network size, the best method for pruning relative to a data set might well differ for the 10% and the 90% limit criteria. As already observed for the final network size, the method of W. Finnoff *et al.* and the smallest weight removal methods perform almost equally good.

Table 5 concerns the generalization performance of the best generalization performance per simulation and gives a similar result as table 4: the number of experiments, for which the $min(\sigma)$ and the $min(\mathbf{w})$ methods produced the neural networks with the best generalization performance, is about equal. But some of the networks produced by the method of E. D. Karnin generalize better than others, and in one case the networks found by the *skeletonization* method do so.

Comparing tables 4 and 5 for whether the best method differs when looking at the generalization performance of the smallest network or for the best generalization performance per simulation, one sees that the best method is in most of the cases equal in the 10% limit but differs often for the 90% limit.

The comparison of the columns with the generalization performance for the networks, which are not pruned, show that the pruning is most of the cases beneficial (independent of the pruning method), if the best network is chosen. This is not the case, if the smallest network is chosen. Then in approximately 29 percent of the experiments for which all types of pruning techniques have been applied, the unpruned network generalizes better (for 44 of 152 ($\approx$ 29%) experiments in table 4 the

	Ω	*min*(σ)	*min*($\mathbf{w}$)	Karnin	*skelet.*	*autoprune*
Solar	2	5	0	2	7	0
	3	5	0	0	–	3
	4	0	0	0	–	1
Wine	2	9	3	11	11	5
Servo	2	6	0	32	0	0
	3	15	26	16	–	25
Auto-mpg	2	0	0	0	90	0
	3	1	0	3	–	0
Glass	1	0	34	19	70	48
	2	1	13	6	2	14
	3	0	25	1	–	22
Monk 1	2	100	100	7	100	100
	3	100	100	4	–	–
Monk 2	2	15	13	10	0	10
	3	59	61	19	–	61
Monk 3	2	39	73	–	–	80
	3	48	55	1	31	58
digits 8x8	<2	0	0	–	–	0

Table 3: Percentage of simulations with the smallest network having the best generalization performance.

unpruned network performed better).

The tables 4 and 5 show that, similarly to the observations made for the final network size, the best pruning method can differ for the 10% and the 90% limit criteria. This is the case in about 20% of the series of experiments performed for this study.

All five pruning methods studied perform on average equally well, but a difference in performance on certain data sets exists: networks pruned by the $min(\sigma)$, the $min(w)$, and the method of M. C. Mozer *et al.* classify all test patterns of the monks 1 data set correctly, whereas the method of E. D. Karnin reaches only 88%. For some data sets with a real-valued target, the mean square error difference for the networks pruned by the five methods for the test set are as high as 25%.

4 Assumptions on the Network Behavior

Although only little of the convergence behavior of neural networks is known *a priori*, one is urged to make assumptions about it in order to have a base for the development of algorithms. Some of these assumptions, albeit seeming almost natural and therefore being often exploited in neural network training algorithms, turn out to be not sound with the results of the experiments. One of those assumptions is the already addressed generalization performance of a neural network which not necessarily increases with a shrinking network size.

Another assumption is that the generalization performance first increases with a shrinking network, and when a certain point is reached, it only decreases. A key point in techniques using this assumption is the detection whether the changes in generalization performance are due to statistical fluctuations or due to the insufficient network size. Some methods therefore trace the development of the generalization performance and re-store the best network found after the training session is terminated. The training session is stopped when the generalization performance degrades for a certain number of connection removals (usually between 3 and 10).

		10% limit for method						90% limit for method						
	ε	Unpruned	*min*(σ)	*min*(**w**)	Karnin	*skelet.*	*autoprune*	Unpruned	*min*(σ)	*min*(**w**)	Karnin	*skelet.*	*autoprune*	
Solar	2	0.078	0.077	0.095	**0.075**	0.089	0.095	0.104	**0.087**	0.103	0.089	0.108	0.103	!!
	3	0.096	0.078	0.087	**0.075**	—	0.084	0.139	0.111	0.105	**0.087**	—	0.105	
	4	0.077	**0.076**	0.077	0.079	—	0.077	0.090	**0.084**	0.092	0.092	—	0.092	
Wine	2	0.122	**0.122**	0.146	**0.122**	**0.122**	**0.122**	0.146	**0.195**	0.220	0.220	**0.195**	**0.195**	!!
Servo	2	0.137	**0.106**	0.123	0.114	0.140	0.123	0.139	**0.110**	0.124	0.124	0.145	0.124	
	3	0.117	0.094	0.105	**0.091**	—	0.105	0.119	0.121	0.116	**0.111**	—	0.116	
Auto-mpg	2	0.055	0.058	0.059	0.058	**0.054**	0.059	0.056	0.059	0.061	0.061	**0.056**	0.061	
	3	0.056	0.058	0.060	**0.057**	—	0.060	0.059	**0.061**	0.062	0.062	—	0.062	!!
Glass	1	0.032	0.033	0.028	0.032	**0.027**	0.028	0.034	0.036	0.032	0.035	**0.028**	0.032	
	2	0.038	0.041	**0.034**	0.037	0.036	**0.034**	0.043	0.046	**0.041**	0.046	0.048	**0.041**	
	3	0.043	0.046	**0.034**	—	—	**0.035**	0.050	0.051	**0.044**	—	—	**0.044**	
Monk 1	2	0.201	**0.000**	**0.000**	0.123	**0.000**	**0.000**	0.218	**0.000**	**0.000**	0.222	**0.000**	**0.000**	
	3	0.275	**0.000**	**0.000**	—	—	—	0.315	**0.000**	**0.000**	—	—	—	
Monk 2	2	0.215	**0.046**	0.093	0.120	0.160	0.093	0.222	**0.116**	0.125	0.206	0.192	0.127	
	3	0.368	**0.160**	**0.169**	0.269	—	**0.169**	0.389	**0.269**	**0.269**	0.326	—	**0.266**	
Monk 3	2	0.097	**0.097**	0.125	—	—	0.123	0.175	**0.144**	0.176	—	—	0.176	
	3	0.074	0.083	**0.074**	0.236	**0.074**	**0.074**	0.234	**0.139**	0.148	0.368	0.167	**0.134**	!!
Digits 8x8	2r	0.076	0.078	**0.074**	—	—	**0.074**	0.100	0.104	**0.098**	—	—	**0.098**	

Table 4: The error on the test set of the smallest network per pruning session in the 10% and the 90% limits.

In order to validate this technique and to determine how many connections to remove per pruning step, the longest decreasing slope (in number of removed connections) of the generalization performance per simulation has to be examined. Then, the number of slopes of a certain length was calculated for all benchmarks and experiments and finally integrated up to a certain length. The result is displayed in figure 5. This plot can be interpreted as a probability to find the network with the optimal generalization performance when the training is aborted after the generalization performance decreased for a certain number of removed connections.

Figure 5 shows for example that if the maximal number of removed connections causing a decrease of the generalization performance is chosen to be 10, the network with an optimal size is found with a probability of 82%. In other words, when the training is stopped if the successive pruning of 10 connections does not lead to an increase of the generalization performance, in 18% of the training sessions, the best performing network is not found. Similarly, if this probability is desired to be at least 90% (95%), this number has to be 22 (47). The longest decreasing slope observed has the length of 226 connections.

5 Complexity of the Methods

As for other algorithms, an important aspect of pruning methods is their computational complexity. If their complexity would not matter, a complete search among all possible neural network topologies could be performed; outranging all other methods in generalization performance and network size. However, this complexity consists not only of the sheer calculation directly related to the decision on which unit to remove, but also the increase in training time due to the required retraining.

Whereas the first item can be calculated for each pruning method, the second is directly related to the *quality*[5] of the pruning method and therefore inaccessible for a theoretical analysis. Although the

[5] Quality in the sense that the pruned network can be retrained to the same performance as the unpruned network in a low number of iterations.

	ω	10% limit for method: Unpruned	*min*(σ)	*min*($\mathbf{w}$)	Karnin	*skelet.*	*autoprune*	90% limit for method: Unpruned	*min*(σ)	*min*($\mathbf{w}$)	Karnin	*skelet.*	*autoprune*	
Solar	2	0.078	0.070	0.070	**0.066**	0.075	0.070	0.104	0.078	0.081	**0.075**	0.092	0.080	
	3	0.096	0.072	0.073	**0.064**	0.094	0.072	0.139	0.082	0.086	**0.073**	0.111	0.083	
	4	0.077	**0.069**	0.070	**0.069**	—	0.070	0.090	**0.072**	0.074	0.073	—	0.074	
Wine	2	0.122	0.098	**0.098**	0.098	0.098	**0.098**	0.146	**0.122**	**0.122**	0.122	0.122	**0.122**	!!
Servo	2	0.137	**0.104**	0.112	0.114	0.137	0.111	0.139	**0.106**	0.114	0.119	0.139	0.114	
	3	0.117	0.092	0.104	**0.091**	—	0.104	0.119	0.109	0.115	**0.106**	—	0.115	
Auto-mpg	2	0.055	0.055	0.055	0.055	**0.054**	0.055	0.056	0.056	**0.056**	0.056	0.056	**0.056**	!!
	3	0.056	0.056	**0.054**	0.056	—	**0.054**	0.059	0.058	**0.056**	0.057	—	**0.056**	
Glass	1	0.032	0.031	0.028	0.032	**0.026**	0.028	0.034	0.032	0.029	0.034	**0.028**	0.030	
	2	0.038	0.037	**0.033**	0.036	0.034	**0.033**	0.043	0.039	**0.036**	0.039	**0.038**	**0.036**	
	3	0.043	0.039	**0.034**	0.038	—	**0.034**	0.050	0.043	**0.039**	0.043	—	**0.039**	
Monk 1	2	0.201	**0.000**	**0.000**	0.069	**0.000**	**0.000**	0.218	**0.000**	**0.000**	0.150	**0.000**	**0.000**	
	3	0.275	**0.000**	**0.000**	0.213	—	—	0.315	**0.000**	**0.000**	0.285	—	—	
Monk 2	2	0.215	**0.019**	0.083	0.104	0.139	0.083	0.222	0.109	**0.102**	0.174	0.141	**0.104**	!!
	3	0.368	**0.157**	**0.169**	0.257	—	**0.164**	0.389	**0.259**	**0.262**	0.315	—	**0.262**	
Monk 3	2	0.097	**0.097**	0.125	—	—	0.120	0.175	**0.130**	0.160	—	—	0.160	
	3	0.074	0.079	**0.074**	0.201	**0.074**	**0.074**	0.234	**0.120**	0.125	0.259	0.130	**0.120**	!!
digits 8x8	2r	0.076	**0.048**	**0.048**	—	—	**0.046**	0.100	0.054	**0.054**	—	—	0.054	

Table 5: The error on the test set of the best performing network per pruning session in the 10% and the 90% limits.

complex methods are already excluded, three groups remain: the $min(\mathbf{w})$ method with a complexity $O(C)$ which is linear to the number C of connections in the network, the $min(\sigma)$ method, the methods of E. D. Karnin and M. C. Mozer *et al.* with $O(C * P)$, and the method of W. Finnoff *et al.* with $O(C * P * I)$ (with I the number of training cycles).

That the re-training time can be important is shown in figure 6: the average retraining time can differ for two pruning heuristics, as it is demonstrated for a second order high order perceptron trained on the Solar data set. After an initial training of about 50 iterations, the first pruning steps require no or only a little retraining, as expected: first the most unimportant connections with the smallest influence on the output are removed. Then the retraining time increases steadily towards the end of the simulation but not equally fast for both methods: the pruning method of Karnin requires on average more retraining per pruning step. It therefore can be concluded that the method of E. D. Karnin is more likely to remove connections which cause an error than the smallest variance method. This coincides with the observation that the method of E. D. Karnin produces on average bigger networks (the dotted curve is terminated before the straight one). Furthermore, it can be seen that very small neural network are expensive to obtain: the number of retraining steps for small high order perceptrons is much higher than for big ones.

6 Conclusions

The pruning of higher order perceptrons has often, but definitively not always, the commonly assumed positive influence on the generalization performance. During a training session, the generalization performance is not increasing steadily while the network decreases in size but shows an unpredictable behavior. Furthermore, in a few experiments, the unpruned ω-th order perceptron generalized on average as well as or even better than the smallest network per simulation. This implies that pruning not necessarily improves the generalization performance of a network. On the contrary, for some data sets the smallest network almost never has the best generalization performance per simulation.

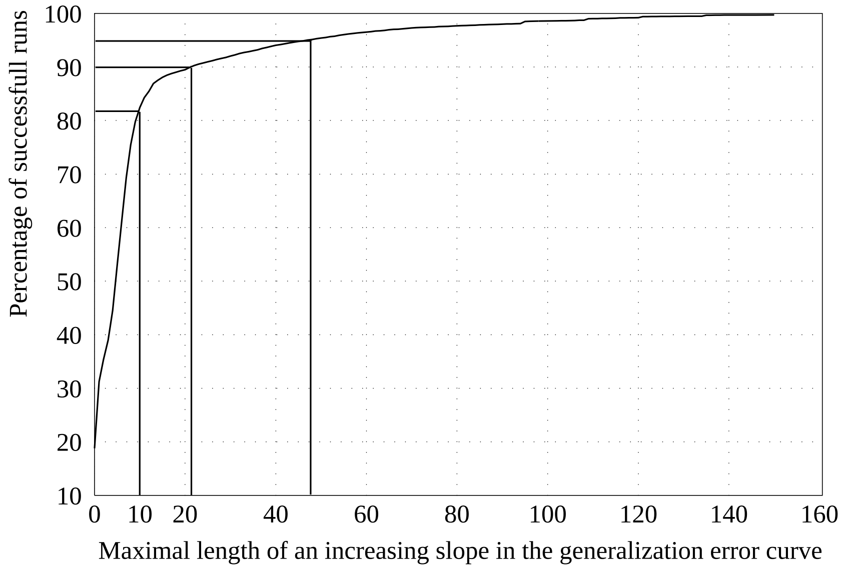

Figure 5: Longest decreasing generalization slopes per simulation.

Therefore, if a network with a good generalization performance is required, networks of a non-optimal size have to be considered.

The method of W. Finnoff *et al.* can be excluded from the set of potential pruning methods, as its performance is similar to the smallest weight removal method, both in terms of network size and generalization performance, but has a higher computational complexity, and its implementation needs more effort.

A significant difference in generalization performance of the final networks produced by the four remaining pruning methods can not be stated generally, only for specific data sets, where the differences are sometimes remarkable. Consequently, if high order perceptrons with a good generalization performance are required, several training sessions using different pruning methods are inevitable, as the best pruning method for a specific data set is *a priori* unknown. It is likely that this observation and its resulting conclusion also apply to other neural network architectures.

The method of E. D. Karnin and the *skeletonization* method are less efficient in finding small high order perceptrons than the $min(w)$ and the $min(\sigma)$ method, independent of the measure used. Furthermore, the latter have the advantage of a lower complexity. Only in a few cases did these methods produce networks of a size comparable to those found by the $min(w)$ and the $min(\sigma)$ method; the final network size is often two or more times as large. The difference between the $min(w)$ and the $min(\sigma)$ method is less significant, with not more than 50% difference in network size. The number of experiments where one of these methods is shown to be more efficient, is comparable.

Pruning methods using penalty terms are excluded from the experiments, as they can force the training process into a “false” minimum which may or may not be close to the global minimum of the error surface (see section 2). Consequently, a neural network should be trained, at least in the final stage of the training, without a penalty term. In other words, regularization techniques should not be used as a pruning technique, but only with the intension of imposing a task related property on the function represented by the trained neural network (for example smoothness of a special weigh on certain training samples).

The in the sections 3.1 and 3.2 introduced 10% and 90% limit measures for the network size and generalization performance show that the performance of two pruning methods can differ remarkably for a certain data set, although the mean generalization performance or the mean final network size is equal for both methods. No pruning method is *a priori* preferred.

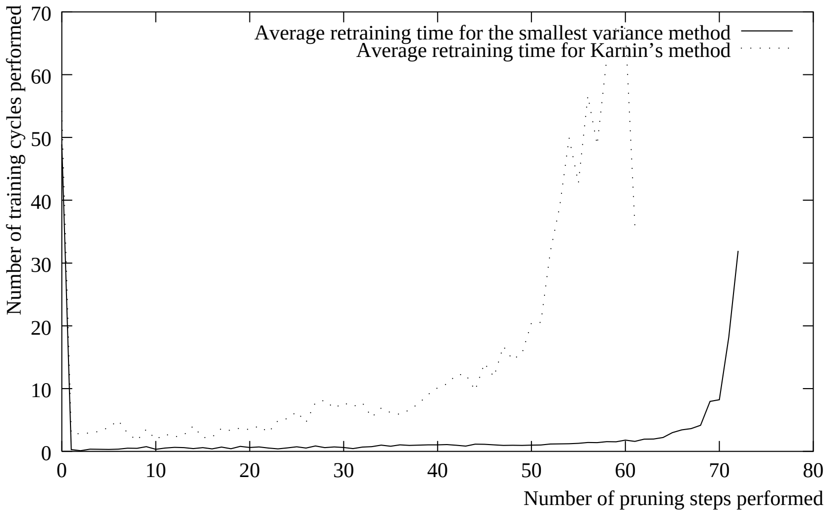

Figure 6: The average number of re-training steps versus pruning step.

A Confidence in Proportions: the 10% and the 90% limit Measure

The confidence in the difference of two sets A and B is judged in the following way: let

$$10\%\ limit = min(10\%\ limit(A), 10\%\ limit(B)),$$

respectively

$$90\%\ limit = min(90\%\ limit(A), 90\%\ limit(B)).$$

Using these limits, the number of elements x_A and x_B smaller or equal than the 10% respectively 90% limit is determined for each set.

Then the confidence in a set having a smaller 10% limit, respectively 90% limit, is defined as the confidence in the difference between two proportions (compare for example [20], pp. 317ff). Consequently, the hypothesis *set A is better in the* 10% *limit, respectively* 90% *limit, measure than B* is tested by rejecting the corresponding null hypothesis. The null hypothesis can be rejected with a confidence of 95%, if

$$1.65 \leq \frac{\frac{x_A}{n_A} - \frac{x_B}{n_B}}{\sqrt{p(1-p)\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}}$$

with $p = \frac{x_A + x_B}{n_A + n_B}$, n_A and n_B the number of elements in set A respectively B, and

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{1.65} e^{\frac{-x^2}{2}} dx \approx 0.95.$$

References

[1] Joydeep Ghosh and Kagan Tumer, "Structural adaptation and generalization in supervised feed-forward networks", *Journal of Artificial Neural Networks*, vol. 1, no. 4, pp. 431–458, 1994.

[2] E. Fiesler and R. Beale (editors), *Handbook of Neural Computation*, Oxford University Press and IOP Publishing, 198 Madison Avenue, New York, NY 10016, 1997.

[3] E. Fiesler, "Minimal and high order neural network topologies", in *Proceedings of the Fifth Workshop on Neural Networks: Academic/Industrial/NASA/Defense; An International Conference on Computational Intelligence: Neural Networks, Fuzzy Systems, Evolutionary Programming and Virtual Reality (WNN93/FNN93)*, San Diego, California, 1993, number 2204 in SPIE Proceedings, pp. 173–178, Simulation Councils, Inc. / The Society for Computer Simulation.

[4] Russell Reed, "Pruning algorithms — a survey", *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, September 1993.

[5] Mike Wynne-Jones, "Constructive algorithms and pruning: Improving the multi layer perceptron", in *Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics*, R. Vichnevetsky and J. J. H. Miller, Eds. IMACS, 1991, vol. 2, pp. 747–750, IMACS International Association for Mathematics and Computers in Simulation(?).

[6] Lutz Prechelt, "PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms", Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, Sept. 1994, Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.Z on ftp.ira.uka.de.

[7] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize", *Neural Networks*, vol. 4, no. 1, pp. 67–69, 1991.

[8] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks", *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239–242, June 1990.

[9] Michael C. Mozer and Paul Smolensky, "Using relevance to reduce network size automatically", *Connection Science*, vol. 1, no. 1, pp. 3–16, 1989.

[10] William Finnoff, Ferdinand Hergert, and Hans Georg Zimmermann, "Improving model selection by nonconvergent methods", *Neural Networks*, vol. 6, pp. 771–783, 1993.

[11] Lutz Prechelt, "Adaptive parameter pruning in neural networks", Tech. Rep. 95–009, International Computer Science Institute, Berkeley, CA, March 1995.

[12] Jean-Luc Beuchat, "Optimisation de réseaux de neurones", report of a student project, supervised by Prof. Nicoud (EPFL, Lausanne, Switzerland) and G. Thimm (IDIAP, Martigny, Switzerland), 1995.

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Flat minima", *Neural Computation*, 1996, accepted for publication.
Also published as "Flat minimum search finds simple nets", Tech. Rep. FKI-200-94, Fakultät für Informatik, Technische Universität München, 80290 München, Germany, December 1994.

[14] Babak Hassibi and David G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon", Tech. Rep. CRC-TR-9214, RICOH California Research Center, Menlo Park, California, USA, May 7, 1992.

[15] Stephen José Hanson and Lorien Y. Pratt, "Comparing biases for minimal network construction with back-propagation", in *Advances in Neural Information Processing Systems 1*, David S. Touretzky, Ed., San Mateo, California, 1989, IEEE, pp. 177–185, Morgan Kaufmann.

[16] Gerald Fahner, Nils Goerke, and Rolf Eckmiller, "Structural adaptation of boolean higher order neurons: Classification with parsimonious topologies for superior generalization", in *Artificial Neural Networks, 2: Proceedings of the 1992 International Conference on Artificial Neural Networks (ICANN-92)*, Igor Alexander and John Taylor, Eds., Amsterdam, The Netherlands, 1992, vol. 1, pp. 285–288, Elsevier Science Publishers B.V. (North-Holland).

[17] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart, "Predicting the future: a connectionist approach", *International Journal of Neural Systems*, vol. 1, no. 3, pp. 193–209, 1990.

[18] Georg Thimm and Emile Fiesler, "High order and multilayer perceptron initialization", *IEEE Transactions on Neural Networks*, vol. 8, no. 2, 1997.

[19] Frederico Girosi, Michael Jones, and Tomaso Poggio, "Regularization theory and neural networks architectures", *Neural Computation*, vol. 7, no. 2, pp. 219–269, March 1995.

[20] John. E. Freund, *Modern Elementary Statistics*, Prentice–Hall, Engelwood Cliffs, New Jersey, forth edition, 1973.