IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

Neural Network Adaptations to Hardware Implementations

Perry Moerland [*] Emile Fiesler

IDIAP–RR 97-17

January 97

PUBLISHED IN

E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*, E1.2:1–13. Institute of Physics Publishing and Oxford University Publishing, New York.

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

[*] e-mail: Perry.Moerland@idiap.ch

IDIAP Research Report 97-17

Neural Network Adaptations to Hardware Implementations

Perry Moerland Emile Fiesler

January 97

published in
E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*, E1.2:1–13. Institute of Physics Publishing and Oxford University Publishing, New York.

Abstract. In order to take advantage of the massive parallelism offered by artificial neural networks, hardware implementations are essential. However, most standard neural network models are not very suitable for implementation in hardware and adaptations are needed. In this section an overview is given of the various issues that are encountered when mapping an ideal neural network model onto a compact and reliable neural network hardware implementation, like quantization, handling non-uniformities and non-ideal responses, and restraining computational complexity. Furthermore, a broad range of hardware-friendly learning rules is presented, which allow for simpler and more reliable hardware implementations. The relevance of these neural network adaptations to hardware is illustrated by their application in existing hardware implementations.

Acknowledgements: The support of the Swiss National Science Foundation (Grant SNSF 21-45621.95) is gratefully acknowledged.

1 Introduction

Soon after the widespread revival of neural network research in the mid-eighties, it was realized that to fully profit from the massive parallelism inherent in neural network models, hardware implementations are essential. This has led to a large variety of implementations using digital and analog electronics, optics, and hybrid techniques. Even though these implementations are largely different, a common denominator is the mapping of neural network algorithms onto reliable, compact, and fast hardware. Any hardware implementation has to optimize three main constraints: accuracy, space, and processing speed. The design of hardware implementations is governed by a balancing of these criteria. An analog implementation, for example, is very efficient in terms of chip area and processing speed, but this comes at the price of a limited accuracy of the network components. In general, this amounts to a trade-off between the accuracy of the implementation and the reliability of its performance. In this section the influence of the limitations typical for hardware implementations will be outlined. Examples of this phenomenon are:

- the quantization of network parameters in digital implementations, in specific its weights, to obtain a far more compact implementation. Its counterpart in analog implementations is a limited accuracy of the network parameters due to system noise.

- and that computation in analog hardware, be it electronic or optical, is characterized by the non-uniformity of its components and by the fact that the components are at best approximations of the corresponding mathematical operations in the neural network model.

This section provides a thorough review of the experimental and theoretical research that has been performed on the behaviour of existing learning algorithms under the limitations imposed by hardware. Furthermore, training algorithms are discussed that offer an improved performance in case of limited accuracy and that further simplify the hardware implementation of neural networks.

In section 2, the effects of a quantization of the network parameters and weight discretization algorithms for various neural network models are reviewed. The different approaches are illustrated with examples from existing neural hardware implementations and several commonly used schemes are discussed in more detail. The influence of hardware non-idealities, such as spatial non-uniformity and non-ideal response is outlined in section 3. Section 4 contains an overview of *hardware-friendly learning algorithms* which are better suited for hardware implementation and especially for on-chip learning. Finally, in section 5, a summary and conclusions are presented.

2 Quantization Effects

The use of very high precision cannot be matched with the goal of developing fast and compact hardware implementations. While in digital implementations a high numerical precision is too area consuming, it is incompatible with the system noise present in analog implementations. Therefore, hardware implementations of neural networks typically use a representation of the network parameters with a limited accuracy. For example, in Philips' L-Neuro 1.0 architecture, which allows the implementation of feedforward networks and on-chip backpropagation training, 16-bit weights are used during the training process and only 4-bit or 8-bit weights are employed during recall [Mauduit-92]. An example of an analog electronic implementation is Intel's Electrically Trainable Analog Neural Network (ETANN), which can perform an impressive two billion weight multiplications per second. The accuracy of its weights and neurons, however, can be compared with a resolution of only seven bits [Holler-89].

Since hardware implementations are characterized by a low numerical precision, it is essential to study its effects on the recall and training of the various neural network models. The need for a further reduction of the accuracy, while retaining a satisfactory network performance, has also led to various *weight discretization* algorithms, especially designed for this purpose. Since most research has been performed for multilayer feedforward networks, these will be discussed separately from the other

Reference	Accuracy	# of Benchmarks		Remarks
	(in bits)	Artificial	Real-world	
[Holt-93]	8	1	-	Finite precision error analysis for the forward retrieving pass.
[Dündar-95]	10	2	-	Statistical model of weight quantization in sigmoidal networks.
[Piché-95]	6–10	2	-	Statistical analysis of the effects of weight errors upon an ensemble of multilayer networks.

Table 1: Weight discretization in multilayer neural networks: off-chip learning.

Reference	Accuracy	# of Benchmarks		Remarks
	(in bits)	Artificial	Real-world	
[Fiesler-88] [Fiesler-90]	2–3	3	-	Forward pass with discrete weights, backward pass with continuous weights.
[Marchesi-93]	3–4	1	1	Power-of-two weights in the forward pass and an adaptive learning rate.
[Tang-93]	3–4	1	-	Power-of-two weights and adaptive gain of the activation function.

Table 2: Weight discretization in multilayer neural networks: chip-in-the-loop learning.

neural network paradigms. A compact overview of a large variety of results on the effects of limited precision in neural networks can be found in Table 1 to 4. These tables list the number of bits that are required for satisfactory (learning) performance and briefly describe the core idea of the algorithms. In order to give an indication of the quality of the experimental evaluation in the cited articles, two columns listing the number of *artificial* and *real-world* benchmarks on which the algorithms have been tested are also included.

2.1 Quantization Effects in Multilayer Neural Networks

Most methods deal with the various aspects of limited precision calculation in multilayer networks. These approaches can be divided into three categories corresponding to the three different training modes for neural network hardware:

Off-chip learning In this case the hardware is not involved in the training process, which is performed on a computer using high precision. The weights resulting from the training process are quantized and then downloaded on the chip. Only the forward propagation pass in the recall phase is performed on-chip which makes these quantization effects amenable for mathematical analysis using a statistical model. Some of the results have been summarized in Table 1 which indicate that the accuracy needed in the on-chip forward pass is around 8 bits. In [Piché-95] a comparison between Heaviside and sigmoidal multilayer networks is given, showing that the weight precision required in a Heaviside network is much higher and even doubles when a layer is added to the network. An interesting practical example illustrating that low on-chip accuracy is sufficient when mapping a neural network trained with a high precision onto a chip, is the application of the analog ANNA chip to high-speed character recognition [Säckinger-92]. Here, a high precision (32-bit floating point) network is mapped on the ANNA chip which uses a 6-bit weight resolution and a 3-bit resolution for the neuron inputs and outputs. The chip's recognition accuracy is only slightly less than the one obtained with floating-point calculations.

Chip-in-the-loop learning In this case the neural network hardware is used during training, but only in forward propagation. The calculation of the new weights is done off-chip on a computer, which downloads the updated weights onto the chip after each training iteration. Several learning algorithms have been proposed that take advantage of the fact that in this way the limited precision only plays a role in the forward propagation pass and that floating-point calculations can be used in the backward pass (Table 2). One of the first, and perhaps most successful, weight discretization techniques is of the chip-in-the-loop kind [Fiesler-88] [Fiesler-90]. It is suitable for feedforward neural networks, easy to implement, and very flexible in that it can handle a large range of discretizations up to the precision of a few bits only (Table 2). The basic idea is to start with a normal neural network with continuous valued weights. These weights are discretized using a staircase shaped *multiple threshold function* and the so created discrete weights are then used for the forward propagation pass of the learning rule. The errors obtained, which are based on the difference between the obtained network outputs and the desired target outputs, are subsequently used to update the continuous valued weights during the backward propagation pass. This scheme is repeated until convergence is obtained. This flexible weight discretization method has been successfully used in the development of the Apple Newton [Lyon-96], and in optical neural networks at Mitsubishi, Japan [Takahashi-91] and in Switzerland [Saxena-95] [Moerland-96.2]. A similar approach has been applied to design neural networks restricted to single power-of-two weights (see section 2.3) [Marchesi-93] [Tang-93].

On-chip learning Here, the training of the neural network is done entirely on-chip which offers the possibility of continuous training. This means in specific that at least the weight values are represented with only a limited precision. Simulations have shown that the popular backpropagation algorithm (see for example [Rumelhart-86]) is highly sensitive to the use of limited precision weights and that training fails when the weight accuracy is lower than 16 bits (first two references in Table 3). This is mainly because the weight updates are often smaller than the quantization step which prevents the weights from changing. In order to reduce the chip area needed for weight storage and to overcome system noise, a further reduction of the number of allowed weight values is desirable. Several weight discretization algorithms have therefore been designed and an extensive list of them and the attainable reduction in required precision is given in Table 3. Some of these weight discretization algorithms have already proven their usefulness in hardware implementations. Battiti's reactive tabu search, for example, has been implemented in the TOTEM processor and successfully applied to a triggering problem in high energy physics with a weight accuracy as low as 4 bits [Battiti-94]. Recently, an analog electronic chip (Kakadu) has been applied successfully to some classification problems by training it with the combined search algorithm and semi-parallel weight perturbation algorithms using only a 6-bit weight accuracy [Jabri-94] [Leong-95].

2.2 Quantization Effects in Other Neural Network Models

Also for other neural network models the effects of a coarse quantization of the weight values on recall and learning have been investigated. The small number of weight discretization algorithms proposed can be partly explained from the fact that the required accuracy for successful learning in these models is lower than for gradient descent learning in multilayer networks (Table 4). An interesting example of a hardware implementation is Bellcore's implementation of a Boltzmann machine and Mean-Field learning, which allows on-chip learning with only 5-bit weights [Alspector-92]. Recently, a weight discretization algorithm for an associative memory with binary $\{-1,+1\}$ weights has been implemented on a digital VLSI chip [Hendrich-96]. The pattern storage capacity that can be obtained with this learning rule is good (0.4 times the number of neurons) and the algorithm is suited for on-chip learning. Verleysen's associative memory training algorithm, that uses the Simplex method to train a network with ternary weights, is best-suited for off-chip training [Verleysen-89].

2.3 Some Remarks on Commonly Used Schemes

A common point of many weight discretization algorithms is the way in which the effects of having only a limited *weight range* are treated. It has been shown by simulations that as soon as the range of the weights decreases below a certain value, which depends on the problem at hand, the training fails to converge because of the clipping of the weight values [Hoehfeld-92]. This can often be solved

Reference	Accuracy	# of Benchmarks		Remarks
	(in bits)	Artificial	Real-world	
[Asanović-91]	16	-	1	Coarse weight quantization in the backpropagation algorithm.
[Holt-93]	14–16	2	-	An error analysis of backpropagation with finite precision.
[Grossman-90]	1	1	-	Adaptation of both weights and the internal representation of the neurons.
[Reyneri-91]	9–10	1	1	Batch backpropagation with a near-optimum learning rate.
[Xie-92]	10	-	2	Weight perturbation with gain adaptation.
[Xie-92]	9	-	2	Combination of weight perturbation and a partial random search.
[Abramson-93]	2	3	-	A slight modification of [Grossman-90] to train sparsely connected Heaviside networks.
[Sakaue-93]	8–10	-	2	A weighted error function in the backpropagation algorithm based on an overestimation of the error.
[Hollis-94]	13	1	-	Weight perturbation with an adaptive gain and learning rate.
[Jabri-94]	6	1	1	Semi-parallel weight perturbation algorithms.
[Simard-94]	16	-	1	Backpropagation without multiplication; gradients and states of power-of-two
[Battiti-95]	1–8	1	2	Heuristic method for solving combinatorial optimization problems.
[Dündar-95]	10	2	-	Backpropagation with forced weight updates.

Table 3: Weight discretization in multilayer neural networks: on-chip learning.

by allowing a dynamic rescaling of the weights (and hence the weight range) by adapting the gain β of the activation function. The calculation of an activation value a_j in a multilayer network is namely done as follows:

$$a_j = \phi(\beta \cdot (\sum_i w_{i,j} \cdot a_i)) \tag{1}$$

Thus, a change of the weight range is equivalent to changing the gain β of the activation function. Various strategies have been proposed to perform this gain adaptation, ranging from heuristics based on the average value of the incoming connections to a neuron [Hoehfeld-92] [Xie-92], to approaches that use some form of gradient descent to train the gains [Tang-93] [Coggins-94].

In some training algorithms the weight values have been limited to powers-of-two [White-92] [Tang-93] [Marchesi-93]. The main advantage of this technique is that all costly multiplications can be replaced by easy to implement shift operations. This scheme has also been applied to gradient values, activation values, and learning rates [Hollis-94] [Simard-94].

Work on limiting the number of weight levels has also been done in the design of Heaviside networks for the computation of boolean functions (majority, parity, comparison, addition) and for the two-spiral problem [Beiu-95] [Beiu-96.1]. Beiu's concern is to minimize the total number of bits required to represent the weights of a network, since this is a realistic measure of the complexity of VLSI implementations. Moreover, it opens up the possibility to compare results obtained by learning algorithms with the entropy (*number of bits*) upper bounds of the data set [Beiu-96.2].

Finally, we would like to point out that a comparative benchmarking study of quantization effects on different neural network models and the improvements that can be obtained by weight discretization algorithms has not yet been done. The accuracies listed in Table 1 to 4 are therefore highly biased by the different benchmarks that were used by the various authors.

Reference	Accuracy	# of Benchmarks		Remarks
	(in bits)	Artificial	Real-world	
Self-organizing map, see for example [Kohonen-89]				
[Kohonen-93]	3–4	-	1	Quantization of input values during recall.
[Rueping-94]	4	2	1	Power-of-two adaptation factor and quantized weights.
[Thiran-94]	5	1	-	Uses a conical neighbourhood function instead of a rectangular one.
Associative memory, see for example [Hopfield-82]				
[Verleysen-89]	2	1	-	A linear programming learning algorithm for associative memories.
[Johannet-92]	9–11	1	-	Integer arithmetics for learning in associative memory.
[Hendrich-96]	1	1	-	Associative memory with binary weights and a good storage capacity.
Boltzmann network [Ackley-85]				
[Balzer-91]	6–8	2	-	Coarse quantization of the weights during learning.
[Alspector-92]	5	2	-	Coarse weight quantization for Boltzmann and mean-field learning.
Neocognitron [Fukushima-80]				
[White-92]	3	1	-	Uses power-of-two weights.
Cascade topology [Fahlman-90]				
[Hoehfeld-92]	12	2	1	Coarse weight quantization in the cascade correlation algorithm.
[Hoehfeld-92]	6	2	1	Cascade correlation with probabilistic rounding and variable gain.
[Campbell-95]	1	2	1	A constructive algorithm for Heaviside cascade networks.

Table 4: Weight discretization in other neural network models.

3 Hardware Non-Idealities

Both in analog electronic and optical neural network implementations, computation suffers from drawbacks which do not play an important role in digital hardware. Some characteristic examples of such non-idealities inherent to analog computation are the spatial non-uniformity of components and non-ideal responses. In this section, examples of these non-idealities are presented, together with their effects on the learning behaviour of neural networks.

3.1 Component Non-Uniformity

Variations between the on-chip components, such as multipliers [Cairns-94] and the read-out of optical weight matrices [Robinson-92], are inevitable in analog hardware. These non-uniformities are particularly troublesome when the training of the network is done off-chip without taking these component variations into account [Frye-91]. It is, however, widely claimed that chip-in-the-loop or on-chip learning can compensate to a considerable extent for these non-uniformities [Card-92]. This is also intuitively clear because the use of the analog circuit in the forward pass incorporates the non-uniformities in the learning process. This has been confirmed by experimental results, for example for on-chip learning in backpropagation networks [Cairns-94] [Dolenko-95]. Their research indicates that backpropagation learning can adapt to the non-uniformity of multiplier gains which are caused by fabrication inaccuracies. The occurrence of additive offsets in the multiplications and especially in weight adaptations do pose serious problems which are not easily overcome by on-chip learning [Dolenko-95]. A possible solution is the use of some dedicated hardware in the weight adaptation circuitry which enables offset-compensation [Annema-95].

3.2 Non-Ideal Response

Computations performed in hardware are approximations of the mathematical operations assumed to be ideal in neural network models. This affects in particular the analog implementation of a linear multiplication and the implementation of a non-linear activation function like the widely-used standard sigmoid. The use of a linear multiplier with a reasonable operating range leads to a large area penalty in VLSI implementations. Therefore, simple non-linear multipliers are often preferable and are used in both electronic [Lont-92] [Hollis-94] [Reyneri-95] and optical implementations [Robinson-92] [Neiberg-94]. The claims on the learning behaviour of a neural network with non-linear multipliers are rather contradictory. While in [Cairns-94] [Dolenko-95] the straightforward use of non-linear multipliers in simulations of on-chip learning in analog backpropagation networks leads to satisfactory results, in [Lont-92] the standard backpropagation algorithm fails to converge with non-linear synapses. Instead, Lont proposes to incorporate non-linear multipliers in the formulation of the backpropagation rule, which leads to good results. A disadvantage of this approach is that an accurate model of the on-chip multiplier is needed. This can be alleviated by *chain-rule perturbation learning* [Hollis-94], which only performs a forward pass through a multilayer network and hence incorporates the hardware characteristics directly into the training. A solution sometimes applied in optical networks is the use of an additional weight mask which complements and thereby compensates for the non-linearities in the multiplier [Neiberg-94].

Another problem for analog hardware is the requisite of an activation function that is similar to the standard sigmoid. The incorporation of a model of a sigmoid-like hardware activation function in the training algorithm can compensate for some inaccuracy [Lont-92]. This is another example of the opportunism that often plays a role in the design of neural hardware: search for the hidden advantages of apparent drawbacks and try to exploit these instead of trying to approximate the existing mathematical model as close as possible. Another approach is the use of a simplified activation function, for example the replacement of the Gaussian function in radial basis networks by a triangular one [Dogaru-96], leading to a simplified hardware implementation. Additional difficulties arise when the activation functions are implemented by optical hardware, as for example liquid crystal light valves. These optical activation functions are characterized, among other non-idealities, by a gain β that differs greatly from the standard value of one, as can be seen in figure 1 where a sigmoid with a gain of approximately 1/161 is depicted [Saxena-95]. While in analog electronics one can try to compensate for a non-standard gain by including a gain stage, this is not possible in optical implementations. In theory one could add additional optical components whose aim would be a modification of the effective gain, but this would increase the complexity and cost of the system, as well as introduce new side effects. A nice and simple way to solve this problem is by using an adapted backpropagation learning rule that is based on a simple and precise relationship between the gain and two other network parameters [Thimm-96], which compensates for a non-standard gain without any additional hardware, and shows superior results [Moerland-95].

4 Hardware-Friendly Learning Algorithms

In this section a variety of learning algorithms that are well suited for hardware implementations of neural networks are presented. These *hardware-friendly learning algorithms* [Moerland-96.1] can de divided into two classes, namely:

1. Adaptations of existing neural network learning rules that facilitate their hardware implementation.

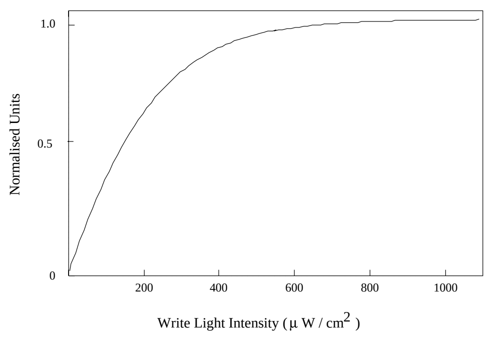

Figure 1: Response curve of an LCLV.

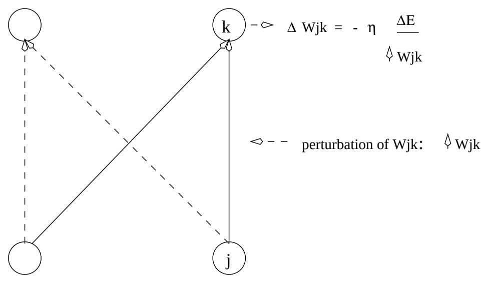

Figure 2: A schematic of the weight perturbation algorithm.

2. Learning algorithms that are by their very conception suitable for hardware implementation.

Here, the emphasis will be on the first of these two classes of hardware-friendly learning algorithms. An example of the second class are *cellular neural networks* which are of special interest for VLSI implementation because of their sparse local connectivity: every unit of the network is a simple analog processor that interacts only with its neighbouring units; see [Chua-93] for a survey. Another example is the class of RAM-based networks which can be easily implemented with standard available components. A recent overview of RAM-based networks and related implementation aspects is given in [Austin-94].

Various hardware-friendlier alternatives have been proposed for several neural network learning rules, especially with the objective to enable *on-chip* learning. The most significant ones are discussed in this section, with an emphasis on hardware-friendly alternatives of the backpropagation algorithm for training multilayer neural networks.

4.1 Perturbation Algorithms

The most popular algorithm for the training of multilayer networks is the backpropagation algorithm, see for example [Rumelhart-86]. However, the realization of large backpropagation networks in analog hardware poses serious problems because of the need for separate or bidirectional circuitry for the backward pass of the algorithm. Other problems are the need of an accurate derivative of the activation function and the cascading of multipliers in the backward pass.

The general idea of perturbation algorithms is to obtain a direct estimate of the gradients by a slight random perturbation of some network parameters, using the forward pass of the network to measure the resulting network error. Thus, these on-chip training techniques do not only eliminate the complex backward pass but are also likely to be more robust to non-idealities occurring in hardware.

The two main variants of this class of algorithms are *node perturbation* which is based on the perturbation of the input value of a neuron, as for example the *Madaline-3* rule [Widrow-90], and *weight perturbation*, see for example [Jabri-92]. The basic concepts of weight perturbation (figure 2) are easily explained by the observation that the gradient descent weight update can be approximated by finite differences ($\upharpoonright W_{jk}$ denotes the perturbation or change of W_{jk}):

$$\Delta W_{jk} = -\eta \cdot \frac{\partial E}{\partial W_{jk}} \approx -\eta \cdot \frac{\Delta E}{\upharpoonright W_{jk}}. \tag{2}$$

The Madaline-3 rule is based on an application of the chain-rule that is standard in the derivation of the backpropagation algorithm (s_k denotes the input to neuron k and $\upharpoonright s_k$ its perturbation):

$$\Delta W_{jk} = -\eta \cdot \frac{\partial E}{\partial W_{jk}} = -\eta \cdot \frac{\partial E}{\partial s_k} \cdot \frac{\partial s_k}{\partial W_{jk}} \approx -\eta \cdot \frac{\Delta E}{\upharpoonright s_k} \cdot a_j. \tag{3}$$

The main disadvantage of these perturbation algorithms is their sequential nature, as opposed to the weight update calculation in the backpropagation algorithm which can, in principle, be performed in parallel. The main differences between the Madaline-3 rule and weight perturbation are the simpler addressing and routing circuitry needed for the latter and the lower computational complexity of the Madaline-3 rule. As can be seen in table 3, weight perturbation also has a good performance with limited precision weights [Xie-92]. Moreover, it is more robust against non-idealities occurring in analog hardware: non-uniformity, non-ideal circuit response, and noise [Cairns-94]. The reason for this is that in this algorithm modeling of activation functions and multipliers does not need to be done, since these form an integral part of the training algorithm. It is interesting to note that the derivation of the Madaline-3 rule does assume the multiplication to be linear which makes possible the reduction of $\partial s_k / \partial W_{jk}$ to a_j in equation 3.

The sequential nature of these simple perturbation algorithms has led to more intricate variants which perform some of the calculations in parallel. A simultaneous perturbation of all weights is a promising alternative [Alspector-93] [Cauwenberghs-93], even when for a reliable estimate of the gradient the results of several perturbations should be averaged or a very small and accurate perturbation is required. Other variants use a semi-parallel perturbation scheme like *chain-rule* perturbation [Hollis-94], *fan-out* or *fan-in-out* perturbation [Jabri-94], and *summed weight neuron* perturbation [Flower-93]. These semi-parallel techniques perturb simultaneously all the weights feeding into or leaving one neuron. An experimental comparison of these perturbation algorithms with an analog multi-layer perceptron chip (Kakadu) in-the-loop showed that the semi-parallel techniques are best suited for effective learning when the accuracy is low [Jabri-94]. The fan-in-out technique showed the best generalization and training convergence results when the weights and weight updates were quantized to 6 bits.

4.2 Local Learning Algorithms

The implementation of a learning rule can be greatly simplified if it only uses information that is locally available [Palmieri-93]. This feature minimizes the amount of wiring and communication. Since the backpropagation algorithm is not local, several local learning algorithms have been designed that avoid a global backpropagation of error signals. An example is an anti-Hebbian learning algorithm that is suitable for optical neural networks [Psaltis-93]. The weight updates in this algorithm depend only on the input and output of that layer and one global error signal. Although it is not a steepest descent rule, it is still guaranteed that the weights are updated in the descent direction. Another local learning rule has been developed in [Brandt-94] which uses only the rates of change of the outgoing weights of a neuron. One of their algorithms is mathematically equivalent to the backpropagation algorithm, but the measurement of the rates of change of the weights could be hard to implement. A promising approach is taken in the Alopex algorithm [Venugopal-91] [Unnikrishnan-94] which is a stochastic algorithm based on the correlation between individual weight changes and changes in the network's error measure. The main advantages of this approach are that the weights can be updated synchronously and that no modeling of the multipliers and activation functions is needed.

4.3 Networks with Heaviside Functions

The design of a compact digital neural network can be simplified considerably when Heaviside functions are used as activation functions instead of a differentiable sigmoidal activation function. While training algorithms for perceptrons with Heaviside functions abound, training multilayer networks with non-differentiable Heaviside functions requires the development of new algorithms. One of the earliest examples of such a learning rule is the *Madaline-2* rule [Widrow-90], which is closely related to the previously described Madaline-3. It is also based on a slight perturbation of the input to a neuron, but in this case the training error is minimized by investigating the effect of an inversion of the activation value of a neuron. If this inversion reduces the Hamming error on the output neurons, the incoming weights of the inverted neuron are adapted with a perceptron training algorithm to reinforce this inversion.

There is also a large variety of *constructive* algorithms which gradually build a Heaviside network by adding neurons and weights [Śmieja-93]. The basis of these algorithms is often formed by a perceptron algorithm that is used to adapt the weights into the freshly added neurons. Recently, some digital and mixed analog/digital architectures have been designed to be suitable for the implementation of a range of these constructive algorithms [Moreno-94].

4.4 Robustness

In section 3 several examples have already been given of the robustness of neural networks to hardware non-idealities. Some research has also been devoted to the robustness of a network to unreliable neurons. This unreliability can consist of sign inversions of hidden neuron values [Judd-93] or destruction of hidden neurons [Kerlirzin-95]. While neural networks trained by standard learning algorithms are not *inherently* fault-tolerant, the incorporation of the expected faults in the training phase leads to remarkable improvements. An illustration of this fact is an adaptation of the backpropagation learning rule that uses only a random subset of hidden neurons for each iteration. The trained network is far more robust to the destruction of hidden neurons and shows performance comparable to the noiseless case [Kerlirzin-95]. This is closely related to the injection of random noise in the weight values during the training of a multilayer neural network, whose effects have been elaborately discussed by Murray and Edwards [Murray-94]. It is demonstrated both analytically and experimentally that this synaptic noise improves the network's fault tolerance to weight damage, generalization to unseen patterns, and training time. Similar results have been obtained when injecting additive noise into the weights of recurrent neural networks [Jim-94].

4.5 Other Hardware-Friendly Neural Network Models

Although the majority of neural hardware is concerned with the implementation of multilayer networks, because of their wide-ranging applicability, most other popular neural network models have also been implemented in hardware. A few examples of the use of hardware-friendly learning in self-organizing feature maps and recurrent networks are given here.

Self-organizing maps One of the requisites of a neural network hardware implementation is the effective use of the processor resources. In general, batch processing is an appropriate alternative to obtain better parallelisation. Kohonen's original algorithm, however, has both an on-line selection of the neuron closest to the input pattern, the *winner neuron*, and an on-line weight update. Two possible variants are to have a batch winner selection combined with either a batch or an on-line weight update. In [Vassilas-95] the convergence properties of these two variants are shown to be comparable with those of the original algorithm.

Recurrent networks Two widely used paradigms for training recurrent networks are Boltzmann Machine learning and Mean Field Theory learning. The parallelism of a potential hardware implementation is seriously hampered by the required asynchronous update of the neurons. Therefore, in both analog [Pujol-94] and optical [Peterson-90] implementations, a synchronous neuron update is used. Another characteristic of the Boltzmann Machine is the use of simulated annealing to gradually increase the gain of a neuron's activation function. In Bellcore's implementation of a Boltzmann Machine this annealing schedule has been replaced by a gradual decrease of additive noise [Alspector-92], while the main idea of Mean Field Theory learning is to replace the annealing strategy by a deterministic approximation.

5 Summary and Conclusions

In this section an overview has been given of a variety of adaptations of neural network learning to enable their successful hardware implementation. These problems can be as general as the effects of a quantization of the network parameters or those of the non-idealities of hardware components. Other problems are more specific for a certain neural network model, like the complications related to the implementation of the backward pass of the standard backpropagation algorithm.

The effects of quantization on a range of neural network models have been outlined, and weight discretization algorithms have been reviewed. These estimations of the required accuracy for well-known learning algorithms and several of the weight discretization algorithms described are already in use in some large-scale hardware implementations. Designers of digital neurocomputers, for example, profit from the fact that the required weight accuracy for backpropagation training is around 16 bits [Mauduit-92]. An example of a successful implementation of a weight discretization algorithm is Battiti's TOTEM-chip which uses a weight accuracy of 4 bits [Battiti-94].

Compared to the state-of-the-art in digital neural network implementations, the design of analog neural network implementations with non-idealities like component non-uniformity, non-ideal responses, and system noise, is still in a more experimental state. Implementations have therefore been limited to small-scale networks [Leong-95] and it is yet to be shown whether reliable large networks can be realized in practice by analog techniques. An important step towards this goal could be the possibility of on-chip learning, since it has been exemplified that neural network models are remarkably robust to hardware non-idealities when these are incorporated in the training of the network. The development of hardware-friendly learning rules that form an alternative for algorithms which are intricate to implement, like the backpropagation algorithm, is therefore essential. The efficacy of perturbation algorithms illustrates the usefulness of this approach and the first implementations using these training algorithms are emerging [Leong-95].

References

[Abramson-93] S. Abramson, D. Saad, and E. Marom. Training a Neural Network with Ternary Weights Using the CHIR Algorithm. *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 997–1000, November 1993.

[Ackley-85] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[Alspector-92] J. Alspector, A. Jayakumar, and S. Luma. Experimental Evaluation of Learning in a Neural Microsystem. *Advances in Neural Information Processing Systems (NIPS91)*, vol. 4, pp. 871–878, Morgan Kaufmann, San Mateo, 1992.

[Alspector-93] J. Alspector, R. Meir, B. Yuhas, and A. Jayakumar. A Parallel Gradient Descent Method for Learning in Analog VLSI Neural Networks. *Advances in Neural Information Processing Systems (NIPS92)*, vol. 5, pp. 836–844, Morgan Kaufmann, San Mateo CA, 1993.

[Annema-95] A. J. Annema and H. Wallinga. Analog Weight Adaptation Hardware. *Neural Processing Letters*, vol. 2, no. 3, pp. 1–4, 1995.

[Asanović-91] K. Asanović and N. Morgan. Experimental Determination of Precision Requirements for Back-Propagation Training of Artificial Neural Networks. *Proceedings of the Second International Conference MicroNeuro'91*, pp. 9–15, (U. Ramacher, U. Rückert, and J. A. Nossek, eds.), München, Germany, October 1991.

[Austin-94] J. Austin. A Review of RAM Based Neural Networks. *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 58–66, Turin, Italy, September 26–28, 1994, ISBN 0-8186-6710-9.

[Balzer-91] W. Balzer, M. Takahashi, J. Ohta, K. Kyuma. Weight Quantization in Boltzmann Machines. *Neural Networks*, vol. 4, pp. 405–409, 1991.

[Battiti-94] R. Battiti and G. Tecchiolli. TOTEM: A Digital Processor for Neural Networks and Reactive Tabu Search. *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 17–25, Turin, Italy, September 26–28, 1994, ISBN 0-8186-6710-9.

[Battiti-95] R. Battiti and G. Tecchiolli. Training Neural Nets with the Reactive Tabu Search. *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1185–1200, September 1995.

[Beiu-95] V. Beiu. VLSI Complexity of Discrete Neural Networks. Gordon and Breach, New York, 1997.

[Beiu-96.1] V. Beiu. Direct Synthesis of Neural Networks. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 257–264, Lausanne, Switzerland, February 12–14, 1996.

[Beiu-96.2] V. Beiu. Entropy Bounds for Classification Algorithms. *Neural Network World*, vol. 6, pp. 497–505, 1996.

[Brandt-94] R. D. Brandt and F. Lin. Supervised Learning in Neural Networks without Explicit Error Back-Propagation. *Proceedings of the Thirty-Second Allerton Conference on Communication, Control, and Computing*, pp. 294–303, Monticello, Illinois, September 28–30, 1994.

[Cairns-94] G. Cairns and L. Tarassenko. Learning with Analogue VLSI MLPs. *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 67–76, Turin, Italy, September 26–28, 1994, ISBN 0-8186-6710-9.

[Campbell-95] C. Campbell and C. Perez Vincente. The Target Switch Algorithm: A Constructive Learning Procedure for Feed-Forward Neural Networks. *Neural Computation*, vol. 7, no. 6, pp. 1245–1264, November 1995.

[Card-92] H. C. Card and C. R. Schneider. Analog CMOS Neural Circuits - In Situ Learning. *International Journal of Neural Systems*, vol. 3, no. 2, pp. 103–124, 1992.

[Cauwenberghs-93] G. Cauwenberghs. A Fast Stochastic Error-Descent Algorithm for Supervised Learning and Optimization. *Advances in Neural Information Processing Systems (NIPS92)*, vol. 5, pp. 244–251, Morgan Kaufmann, San Mateo CA, 1993.

[Chua-88] L. O. Chua and L. Yang. Cellular Neural Networks: Theory. *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1257–1272, 1988.

[Chua-93] L. O. Chua and T. Roska. The CNN Paradigm. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, volume 40, no. 3, pp. 147–156, March 1993.

[Coggins-94] R. Coggins and M. Jabri. Wattle: A Trainable Gain Analogue VLSI Neural Network. *Advances in Neural Information Processing Systems (NIPS93)*, vol. 6, pp. 874-881, Morgan Kaufman, San Mateo CA, 1994.

[Dogaru-96] R. Dogaru, A. T. Murgan, S. Ortmann, and M. Glesner. A Modified RBF Neural Network for Efficient Current-Mode VLSI Implementation. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 265–270, Lausanne, Switzerland, February 12–14, 1996.

[Dolenko-95] B. K. Dolenko and H. C. Card. Tolerance to Analog Hardware of On-Chip Learning in Backpropagation Networks. *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1045–1052, September 1995.

[Dündar-95] G. Dündar and K. Rose. The Effects of Quantization on Multilayer Neural Networks. *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1446–1451, November 1995.

[Fahlman-90] S. E. Fahlman and C. Lebiere. The Cascade-Correlation Learning Architecture. *Advances in Neural Information Processing Systems (NIPS89)*, vol. 2, pp. 524–532, Morgan Kaufmann, San Mateo CA, 1990.

[Fiesler-88] E. Fiesler, A. Choudry, and H. J. Caulfield. Weight Discretization in Backward Error Propagation Neural Networks. *Neural Networks*, special supplement with "Abstracts of the First Annual INNS Meeting", vol. 1, p. 380, 1988.

[Fiesler-90] E. Fiesler, A. Choudry, and H. J. Caulfield. A Weight Discretization paradigm for Optical Neural Networks. *Proceedings of the International Congress on Optical Science and Engineering*, vol. SPIE 1281, pp. 164–173, SPIE, Bellingham, Washington, 1990, ISBN: 0-8194-0328-8.

[Flower-93] B. Flower and M. Jabri. Summed Weight Neuron Perturbation: An $O(N)$ Improvement over Weight Perturbation. *Advances in Neural Information Processing Systems (NIPS92)*, vol. 5, 212–219, Morgan Kaufmann, San Mateo CA, 1993.

[Frye-91] R. C. Frye, E. A. Rietman, and C. C. Wong. Back-Propagation Learning and Nonidealities in Analog Neural Network Hardware. *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 110–117, January 1991.

[Fukushima-80] K. Fukushima. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[Grossman-90] T. Grossman. The CHIR Algorithm for Feedforward Networks with Binary Weights. *Advances in Neural Information Processing Systems (NIPS89)*, vol. 2, pp. 516–523, Morgan Kaufmann, San Mateo CA, 1990.

[Hendrich-96] N. Hendrich. A Scalable Architecture for Binary Couplings Attractor Neural Networks. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 117–124, Lausanne, Switzerland, February 12–14, IEEE Computer Society Press, Los Alamitos, CA USA, 1996.

[Hoehfeld-92] M. H. Hoehfeld and S. Fahlman. Learning with Limited Numerical Precision Using the Cascade-Correlation Algorithm. *IEEE Transactions on Neural Networks*, vol. 3, no. 4, July 1992.

[Holler-89] M. Holler, S. Tam, H. Castro, and R. Benson. An Electrically Trainable Artificial Neural Network (ETANN) with 10240 'Floating Gate' Synapses. *Proceedings of the International Joint Conference on Neural Networks (IJCNN89)*, vol. 2, pp. 191–196, Washington DC, 1989.

[Hollis-94] P. W. Hollis and J. J. Paulos. A Neural Network Learning Algorithm Tailored for VLSI Implementation. *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 784–791, September 1994.

[Holt-93] J. L. Holt and J.-N. Hwang. Finite Error Precision Analysis of Neural Network Hardware Implementations. *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 1380–1389, March 1993.

[Hopfield-82] J. J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences USA*, vol. 79, no. 8, pp. 2554–2558, Washington DC, April 1982.

[Jabri-92] M. Jabri and B. Flower. Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks. *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 154–157, January 1992.

[Jabri-94] Practical Performance and Credit Assignment Efficiency of Analog Multi-Layer Perceptron Perturbation Based Training Algorithms. SEDAL Technical Report 1-7-94, Systems Engineering and Design Automation Laboratory, Sydney University Electrical Engineering, NSW 2006 Australia, 1994.

[Jim-94] K. Jim, C. L. Giles, and B. G. Horne. Synaptic Noise in Dynamically-Driven Recurrent Neural Networks: Convergence and Generalization. Technical report UMIACS-TR-94-89 / CS-TR-3322, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 USA, May 1994.

[Johannet-92] A. Johannet, L. Personnaz, G. Dreyfus, J.-D. Gascuel, and M. Weinfeld. Specification and Implementation of a Digital Hopfield-Type Associative Memory with On-Chip Training. *IEEE Transactions on Neural Networks*, volume 3, number 4, pp. 529–539, July 1992.

[Judd-93] S. Judd and P. W. Munro. Nets with Unreliable Hidden Nodes Learn Error-Correcting Codes. *Advances in Neural Information Processing Systems (NIPS92)*, vol. 5, pp. 89–96, Morgan Kaufmann, San Mateo CA, 1993.

[Kerlirzin-95] P. Kerlirzin and P. Réfrégier. Theoretical Investigation of the Robustness of Multilayer Perceptrons: Analysis of the Linear Case and Extension to Nonlinear Networks. *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 560–571, May 1995.

[Kohonen-89] T. Kohonen. *Self-Organization and Associative Memory*, 3rd edition, Springer Verlag, Berlin, 1989.

[Kohonen-93] T. Kohonen. Things You Haven't Heard about the Self-Organizing Map. *Proceedings of the 1993 IEEE International Conference on Neural Networks*, vol. 3, pp. 1147–1156, San Francisco, California, March 28–April 1, 1993, ISBN: 0-7803-0999-5.

[Leong-95] P. H. W. Leong and M. A. Jabri. A Low-Power VLSI Arrhythmia Classifier. *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1435–1445, November 1995.

[Lont-92] J. Lont and W. Guggenbühl. Analog CMOS Implementation of a Multilayer Perceptron with Nonlinear Synapses. *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 385-392, May 1992.

[Lyon-96] R. F. Lyon and L. S. Yaeger. On-Line Hand-Printing Recognition with Neural Networks. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 201–212, Lausanne, Switzerland, February 12–14, 1996.

[Marchesi-93] M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini. Fast Neural Networks Without Multipliers. *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 53–62, January 1993.

[Mauduit-92] N. Mauduit, M. Duranton, J. Gobert, and J.-A. Sirat. Lneuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems. *IEEE Transactions on Neural Networks*, volume 3, number 3, pages 414–422, May 1992.

[Moerland-95] P. Moerland, E. Fiesler, and I. Saxena. The Effects of Optical Thresholding in Backpropagation Neural Networks. *Proceedings of the International Conference on Artificial Neural Networks (ICANN95)*, vol. 2, pp. 339–343, Paris, France, October 9–13, 1995.

[Moerland-96.1] P. Moerland and E. Fiesler. Hardware-Friendly Learning Algorithms for Neural Networks: An Overview. *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 117–124, Lausanne, Switzerland, February 12–14, 1996.

[Moerland-96.2] P. Moerland, E. Fiesler and I. Saxena. Discrete All-Positive Multilayer Perceptrons for Optical Implementations. IDIAP-RR 97-02, IDIAP, Martigny, Switzerland, February 1997 (accepted for publication in *Optical Engineering*).

[Moreno-94] J. M. Moreno Arostegui. VLSI Architectures for Evolutive Neural Models. Ph.D. Thesis, Technical University of Catalunya, Department of Electronics Engineering, Barcelona, Spain, 1994.

[Murray-94] A. F. Murray and P. J. Edwards. Enhanced MLP Performance and Fault Tolerance Resulting from Synaptic Weight Noise During Training. *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 792–802, September 1994.

[Neiberg-94] L. Neiberg and D. Casasent. High-Capacity Neural Networks on Nonideal Hardware. *Applied Optics*, vol. 33, no. 32, pp. 7665–7675, 1994.

[Palmieri-93] F. Palmieri, J. Zhu, and C. Chang. Anti-Hebbian Learning in Topologically Constrained Linear Networks: A Tutorial. *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 748–761, September 1993.

[Peterson-90] C. Peterson, S. Redfield, J. D. Keeler, and E. Hartman. An Optoelectronic Architecture for Multilayer Learning in a Single Photorefractive Crystal. *Neural Computation*, vol. 2, pp. 25–34, 1990.

[Piché-95] S. W. Piché. The Selection of Weight Accuracies for Madalines. *IEEE Transactions on Neural Networks*, vol. 6, no. 2, p. 432–445, March 1995.

[Protzel-93] P. W. Protzel, D. L. Palumbo, and M. K. Arras. Performance and Fault-Tolerance of Neural Networks for Optimization. *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 600–614, July 1993.

[Psaltis-93] D. Psaltis and Y. Qiao. Adaptive Multilayer Optical Networks, In *Progress in Optics*, editor: E. Wolf, vol. 31, chapter 4, pp. 227–261, 1993, Elsevier Science Publishers, Amsterdam, The Netherlands, ISBN 0–444–89836–0.

[Pujol-94] H. Pujol, O. Klein, E. Belhaire, and P. Garda. RA: An Analog Neurocomputer for the Synchronous Boltzmann Machine. *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 449–455, Turin, Italy, September 26–28, 1994, ISBN 0-8186-6710-9.

[Reyneri-91] L. M. Reyneri and E. Filippi. An Analysis on the Performance of Silicon Implementations of Backpropagation Algorithms for Artificial Neural Networks. *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1380–1389, December 1991.

[Reyneri-95] L. M. Reyneri. A Performance Analysis of Pulse Stream Neural and Fuzzy Computing Systems. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 42, no. 10, pp. 642–660, October 1995.

[Robinson-92] M. G. Robinson and K. M. Johnson. Noise Analysis of Polarization-Based Optoelectronic Connectionist Machines. *Applied Optics*, vol. 31, no. 2, pp. 263–272, January 1992.

[Rueping-94] S. Rueping, K. Goser, and U. Rueckert. A Chip for Selforganizing Feature Maps. *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 26–33, Turin, Italy, September 26–28, 1994, ISBN 0-8186-6710-9.

[Rumelhart-86] D. Rumelhart, G. Hinton, and R. Williams. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations, pp. 318–362, MIT Press, Cambridge, Massachusetts, 1986, ISBN: 0–262–18210–7.

[Säckinger-92] E. Säckinger, B.E. Boser, J. Bromley, Y. LeCun, and L. D. Jackel. Application of the ANNA Neural Network Chip to High-Speed Character Recognition. *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 498–505, May 1992.

[Sakaue-93] S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, and Y. Shimeki. Reduction of Required Precision Bits for Backpropagation Applied to Pattern Recognition. *IEEE Transactions on Neural Networks*, vol. 4, no. 2, March 1993.

[Saxena-95] I. Saxena and E. Fiesler. Adaptive Multilayer Optical Neural Network with Optical Thresholding. *Optical Engineering* (ISSN 0091–3286), special on optics in Switzerland (P. Rastogi, editor), vol. 34 , no. 8, pp. 2435–2440, August 1995.

[Simard-94] P. Y. Simard and H. P. Graf. Backpropagation without Multiplication. *Advances in Neural Information Processing Systems (NIPS93)*, (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), vol. 6, pp. 232–239, Morgan Kaufman, San Mateo CA, 1993.

[Śmieja-93] F. J. Śmieja. Neural Network Constructive Algorithms: Trading Generalization for Learning Efficiency? *Circuits, Systems, and Signal Processing*, vol. 12, no. 2, pp. 331–374, 1993.

[Takahashi-91] M. Takahashi, M. Oita, S. Tai, K. Kojima, and K. Kyuma. A Quantized Back Propagation Learning Rule and its Application to Optical Neural Networks. *Optical Computing and Processing; The Science and Technology of Optics in Computing, Communications, Switching and Information Processing*, vol. 1, no. 2, pp. 175–182, 1991.

[Tang-93] C. Z. Tang and H. K. Kwan. Multilayer Feedforward Neural Networks with Single Power-of-Two Weights. *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2724–2727, August 1993.

[Thimm-96] G. Thimm, P. Moerland, and E. Fiesler. The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks, *Neural Computation*, vol. 8, no. 2, pp. 251–260, February 1996.

[Thiran-94] P. Thiran, V. Peiris, P. Heim, and B. Hochet. Quantization Effects in Digitally Behaving Circuit Implementations of Kohonen Networks. *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 450–458, May 1994.

[Unnikrishnan-94] K. P. Unnikrishnan and K. P. Venugopal. Alopex: A Correlation-Based Learning Algorithm for Feedforward and Recurrent Neural Networks. *Neural Computation*, vol. 6, no. 3, pp. 469, May 1994.

[Vassilas-95] N. Vassilas, P. Thiran, and P. Ienne. How to Modify Kohonen's Self-Organizing Feature Maps for An Efficient Digital Parallel Implementation. *Proceeding of the International Conference on Artificial Neural Networks*, Cambridge, June 26–28, 1995.

[Venugopal-91] K. P. Venugopal and A. S. Pandya. Alopex Algorithm for Training Multilayer Neural Networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Singapore, November, 1991.

[Verleysen-89] M. Verleysen, B. Sirletti, A. Vandemeulebroecke, and P. G. A. Jespers. A High-Storage Capacity Content-Addressable Memory and Its Learning Algorithm. *IEEE Transactions on Circuits and Systems*, vol. 36, no. 5, pp. 762-766, May 1989.

[White-92] B. A. White and M. I. Elmasry. The Digi-Neocognitron: A Digital Neocognitron Neural Network Model for VLSI. *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 73–85, January 1992.

[Widrow-90] B. Widrow and M. A. Lehr. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proceedings of the IEEE*, vol. 78, no. 9, 1415–1442, September 1990.

[Xie-92] Y. Xie and M. A. Jabri. Training Limited Precision Feedforward Neural Networks. *Proceedings of the 3rd Australian Conference on Neural Networks*, pp. 68–71, 1992.