The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks *

G. Thimm, P. Moerland, and E. Fiesler

IDIAP

P. O. Box 592
CH-1920 Martigny, Switzerland
Electronic mail: Thimm@IDIAP.CH

Abstract

The backpropagation algorithm is widely used for training multilayer neural networks. In this publication the gain of its activation function(s) is investigated. In specific, it is proven that changing the gain of the activation function is equivalent to changing the learning rate and the weights. This simplifies the backpropagation learning rule by eliminating one of its parameters. The theorem can be extended to hold for some well-known variations on the backpropagation algorithm, such as using a momentum term, flat spot elimination, or adaptive gain. Furthermore, it is successfully applied to compensate for the non-standard gain of optical sigmoids for optical neural networks.

Keywords: *neural network, neural computation, neural computing, connectionism, neurocomputing, multilayer neural network, backpropagation, (sigmoid) steepness, gain, slope, temperature, adaptive gain, (steep) activation function, (adaptive) learning rate, initial weight, momentum, flat spot elimination, weight discretization, threshold, bias, optical implementation.*

1 Introduction

When using the backpropagation algorithm[1] to train a multilayer neural network, one is free to choose parameters like the initial weight distribution, learning rate, activation function, network topology, and gain of the activation function.

A common choice for the activation function φ of a neuron in a multilayer neural network is the logistic or sigmoid function:

$$\varphi(x) = \frac{\gamma}{1 + e^{-\beta x}}, \tag{1}$$

which has a range $(0, \gamma)$. Alternative choices are a hyperbolic tangent, $\gamma \tanh(\beta x)$, yielding output values in the range $(-\gamma, \gamma)$ and a Gaussian function $\gamma e^{-(\beta x)^2}$ with range $(0, \gamma]$. The parameter β is called the *gain*, and $\gamma\beta$ the *steepness* (slope) of the activation function[2]. The effect of changing the gain of an activation function is illustrated in figure 1: the gain scales the activation function in the direction of the horizontal axis.

This publication proves that a relationship between gain, learning rate, and weights in backpropagation neural networks exists; followed by the implications of this relationship for variations of the backpropagation algorithm. Finally, a direct application of the relationship to the implementation of neural networks with optical activation functions with a non-standard gain is presented.

*This paper is submitted to *Neural Computation*

[1] See for example [Rumelhart-86].

[2] Note that gain and steepness are identical for activation functions with $\gamma = 1$ [Saxena-94]. The term *temperature* is sometimes used as a synonym for the reciprocal of gain.

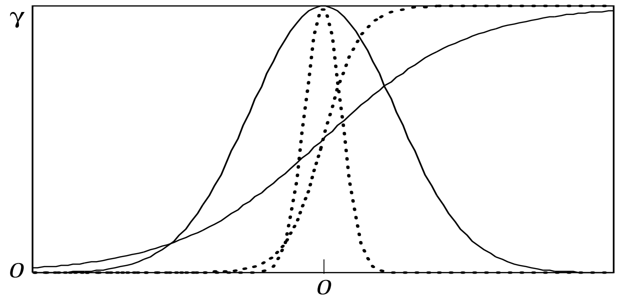

Figure 1: A logistic and a Gaussian function of gain one (solid lines) and their four times steeper counterparts (dotted lines).

Several other authors hypothesized about the existence of a relationship between the gain of the activation function and the weights [Codrington-94] [Wessels-92][3], or between the gain and the learning rate [Kruschke-91] [Mundie-92] [Zurada-92]. In specific, J. Zurada writes: "*... leads to the conclusion that using activation functions with large* λ {=gain (the authors)} *may yield results similar as in the case of large learning constant* η*. It thus seems advisable to keep* λ *at a standard value of 1, and to control learning speed using solely the learning constant* η *...* ".

2 The Backpropagation Algorithm

In this section a generalization of the on-line backpropagation learning rule [Rumelhart-86] is described, in which every neuron has its own (local) learning rate. The standard case of a unique learning rate corresponds to all local learning rates being equal for the whole network.

The following notation and nomenclature is used [Fiesler-94.1]: a (multilayer) neural network can have an arbitrary number of layers denoted by L. The number of neurons in layer ℓ $(1 \leq \ell \leq L)$ is denoted by N_ℓ and the neurons in layer ℓ are numbered from 1 up to N_ℓ. Layer 1 is the input layer[4] and layer L is the output layer of the network. Adjacent layers are fully interlayer connected. The weight from neuron i to neuron j in the next layer ℓ is denoted by $w_{\ell,i,j}$. The activation value of this neuron is indicated as $a_{\ell,j}$ $(j > 0)$, and t_j denotes the target pattern value for output neuron j. To simplify the notation the convention is used that $a_{\ell-1,0} = 1$ and the bias (or offset) is $w_{\ell,0,j}$. The backpropagation algorithm is described by the following five steps:

1. **Initialization** Weights and biases are initialized with random values[5].

2. **Pattern presentation** An input pattern, which is used to initialize the activation values of the neurons in the input layer, and its corresponding target pattern are presented.

3. **Forward propagation** During this phase, the activation values of the input neurons are propagated layerwise through the network. The new activation value of neuron j in layer ℓ $(2 \leq \ell \leq L)$ is

$$a_{\ell,j} = \begin{cases} 1 & \text{if } j = 0 \\ \varphi_{\ell,j}(\mathrm{net}_{\ell,j}) & \text{otherwise,} \end{cases} \tag{2}$$

where the input of a neuron j, not in the input layer, is defined as

$$\mathrm{net}_{\ell,j} = \sum_{i=0}^{N_{\ell-1}} w_{\ell,i,j} a_{\ell-1,i}, \tag{3}$$

where $\varphi_{\ell,j}$ is a differentiable activation function, for example the logistic function (1).

[3] L. Wessels et al. use a weight initialization method which scales the initial weight range according to the gain of the activation function.

[4] A layer is defined here as a layer of neurons and the input layer is counted as number one [Fiesler-94.1].

[5] See [Thimm-94] for an in-depth study of weight initialization.

4. **Backward propagation** For each neuron an error signal δ is calculated, starting at the output layer and then propagating it back through the network:

$$\begin{array}{rcll} \delta_{L,j} & = & \varphi'_{L,j}(\mathrm{net}_{L,j})\,(t_j - a_{L,j}) & \text{for the output layer } L \\ \delta_{\ell,j} & = & \varphi'_{\ell,j}(\mathrm{net}_{\ell,j})\,\sum_k \delta_{\ell+1,k}\,w_{\ell+1,j,k} & \text{for layers } 2 \le \ell \le L-1. \end{array} \tag{4}$$

After the calculation of all error signals, the weights and biases are updated:

$$w_{\ell,i,j} := w_{\ell,i,j} + \eta_{\ell,j}\delta_{\ell,j}a_{\ell-1,i}, \tag{5}$$

where $\eta_{\ell,j}$ denotes the learning rate of neuron j in layer ℓ.

5. **Convergence test** If no convergence go to **Pattern presentation**.

3 The Relationship Between the Gain of the Activation Function, the Learning Rate, and the Weights

The theorem below gives a precise relationship between gain, initial weights, and learning rate for two backpropagation neural networks with the same topology and corresponding neurons have activation functions φ and $\bar{\varphi}$:

$$\varphi_{\ell,j}(x) = \bar{\varphi}_{\ell,j}(\beta_{\ell,j}\,x). \tag{6}$$

This means that corresponding neurons in the two networks have the same activation function, except that the first has gain 1 and the second, gain $\beta_{\ell,j}$.

Theorem 1 Two neural networks M and N of identical topology whose activation function, gain, learning rate, and weights are related to each other as given in table 1, are equivalent under the on-line back propagation algorithm; that is, when presented the same pattern set in the same order, their outputs are identical.

	Network M	Network N
Activation function	$\varphi_{\ell,j}(x) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\,x)$	$\overline{\varphi}_{\ell,j}(x)$
Gain	$\beta_{\ell,j}$	$\bar{\beta}_{\ell,j} = 1$
Learning rate	$\eta_{\ell,j}$	$\bar{\eta}_{\ell,j} = \beta^2_{\ell,j}\,\eta_{\ell,j}$
Weights	$w_{\ell,i,j}$	$\bar{w}_{\ell,i,j} = \beta_{\ell,j}\,w_{\ell,i,j}$

Table 1: The relationship between activation function, gain, weights, and learning rate.

Proof To simplify the notation, the vector of incoming weights of neuron j is denoted by $\mathbf{w}_{\ell,j}$ and the vector of activation values of layer ℓ by $\mathbf{a}_\ell$. Now, using this notation, (3) can be rewritten as:

$$\mathrm{net}_{\ell,j} = \mathbf{w}_{\ell,j} \cdot \mathbf{a}_{\ell-1}, \tag{7}$$

where "$\cdot$" is the inner product operator. The variables of network N (the network with activation functions with gain one) are over lined; like for example $\overline{\mathrm{net}}_{\ell,j} = \overline{\mathbf{w}}_{\ell,j} \cdot \overline{\mathbf{a}}_{\ell-1}$.

The proof of theorem 1 is separated into two parts. Lemma 1 deals with the forward propagation: the networks have the same output for the same input pattern. Lemma 2 deals with the backward propagation: the conditions for lemma 1 still hold after a training step.

Lemma 1 Two networks M and N, satisfying the preconditions given in theorem 1, have the same activation values for corresponding neurons, that is $\mathbf{a}_\ell = \overline{\mathbf{a}}_\ell$ (for all ℓ), if $\mathbf{a}_1 = \overline{\mathbf{a}}_1$ (is forward propagated).

Proof By induction on the number of layers, starting at the input layer.

Induction base: The activation values of the input layer neurons of network M and N are identical, since the input patterns are identical ($\overline{\mathbf{a}}_\ell = \mathbf{a}_\ell$).

Induction step: For neuron j, not in the input layer:

$$\begin{array}{lll}
 & a_{\ell,j} = \overline{a}_{\ell,j} & \text{using (2), trivially fulfilled for } j=0. \\
\Leftrightarrow & \varphi_{\ell,j}(\mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\overline{\mathrm{net}}_{\ell,j}) & \text{using (6): } \varphi_{\ell,j}(x) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j} x) \\
\Leftrightarrow & \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\ \mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\overline{\mathrm{net}}_{\ell,j}) & \\
\Leftarrow & \beta_{\ell,j}\ \mathrm{net}_{\ell,j} = \overline{\mathrm{net}}_{\ell,j} & \text{using (7)} \\
\Leftrightarrow & \beta_{\ell,j}(\mathbf{w}_{\ell,j} \cdot \mathbf{a}_{\ell-1}) = \overline{\mathbf{w}}_{\ell,j} \cdot \overline{\mathbf{a}}_{\ell-1} & \text{on account of } \overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\mathbf{w}_{\ell,j} \\
\Leftrightarrow & \beta_{\ell,j}(\mathbf{w}_{\ell,j} \cdot \mathbf{a}_{\ell-1}) = (\beta_{\ell,j}\mathbf{w}_{\ell,j}) \cdot \overline{\mathbf{a}}_{\ell-1}, &
\end{array}$$

which is true on account of the induction hypothesis that the activation values in the lower layers are identical. Note that in the course of the proof it has also been shown that $\beta_{\ell,j}\ \mathrm{net}_{\ell,j} = \overline{\mathrm{net}}_{\ell,j}$. □

In the proof of Lemma 1 the property $\overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\mathbf{w}_{\ell,j}$ is used. Since the backward propagation changes the weights, it has to be shown that this property is an invariant of the backward propagation step.

Lemma 2 Consider networks M and N, with $\overline{a}_{\ell,j} = a_{\ell,j}$ and $\overline{\mathrm{net}}_{\ell,j} = \beta_{\ell,j}\mathrm{net}_{\ell,j}$ (for all ℓ and j), then

$$\forall j, \ell : \overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\mathbf{w}_{\ell,j} \tag{8}$$

is invariant under the backward propagation step (if the same input and target patterns are propagated through the networks).

Proof Let $\Delta\mathbf{w}_{\ell,j}$ denote the weight change $\eta_{\ell,j}\delta_{\ell,j}\mathbf{a}_{\ell-1}$. One observes that (8) holds if and only if $\Delta\overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\Delta\mathbf{w}_{\ell,j}$ (for all j and ℓ). Manipulating this expression:

$$\begin{array}{lll}
 & \Delta\overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\Delta\mathbf{w}_{\ell,j} & \text{definition of } \Delta\mathbf{w}_{\ell,j} \\
\Leftrightarrow & \overline{\eta}_{\ell,j}\overline{\delta}_{\ell,j}\overline{\mathbf{a}}_{\ell-1} = \beta_{\ell,j}\eta_{\ell,j}\delta_{\ell,j}\mathbf{a}_{\ell-1} & \text{since } \overline{\eta}_{\ell,j} = \beta^2_{\ell,j}\eta_{\ell,j} \text{ and } \forall\ell :\ \overline{\mathbf{a}}_\ell = \mathbf{a}_\ell \text{ (lemma 1)} \\
\Leftarrow & \beta^2_{\ell,j}\eta_{\ell,j}\overline{\delta}_{\ell,j} = \beta_{\ell,j}\eta_{\ell,j}\delta_{\ell,j}, &
\end{array}$$

hence, it needs to be shown that $\beta_{\ell,j}\overline{\delta}_{\ell,j} = \delta_{\ell,j}$, which is done by an induction on the number of layers, starting at the output layer.

Induction base: For the activation values of the output layer the equation $\beta_{L,j}\overline{\delta}_{L,j} = \delta_{L,j}$ holds:

$$\begin{array}{lll}
 & \beta_{L,j}\overline{\delta}_{L,j} = \delta_{L,j} & \text{using (4)} \\
\Leftrightarrow & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\overline{\mathrm{net}}_{L,j})\ (\overline{t}_j - \overline{a}_{L,j}) = \varphi'_{L,j}(\mathrm{net}_{L,j})\ (t_j - a_{L,j}) & \text{since } \overline{a}_{L,j} = a_{L,j},\ \overline{t}_j = t_j \\
\Leftarrow & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\overline{\mathrm{net}}_{L,j}) = \varphi'_{L,j}(\mathrm{net}_{L,j}) & \text{using } \overline{\mathrm{net}}_{L,j} = \beta_{L,j}\mathrm{net}_{L,j} \\
\Leftrightarrow & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\beta_{L,j}\mathrm{net}_{L,j}) = \varphi'_{L,j}(\mathrm{net}_{L,j}), &
\end{array}$$

which follows from applying the chain rule to $\varphi_{\ell,j}(\mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\mathrm{net}_{\ell,j})$.

Induction step: For a neuron j not in the output layer ($\ell < L$):

$$\begin{array}{lll}
 & \beta_{\ell,j}\overline{\delta}_{\ell,j} = \delta_{\ell,j} & \text{using (4)} \\
\Leftrightarrow & \beta_{\ell,j}\ \overline{\varphi}'_{\ell,j}(\overline{\mathrm{net}}_{\ell,j})\ \sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} & \\
 & = \varphi'_{\ell,j}(\mathrm{net}_{\ell,j})\ \sum_k \delta_{\ell+1,k}w_{\ell+1,j,k} & \text{using } \overline{\mathrm{net}}_{\ell,j} = \beta_{\ell,j}\mathrm{net}_{\ell,j} \\
\Leftrightarrow & \beta_{\ell,j}\ \overline{\varphi}'_{\ell,j}(\beta_{\ell,j}\mathrm{net}_{\ell,j})\ \sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} & \\
 & = \varphi'_{\ell,j}(\mathrm{net}_{\ell,j})\ \sum_k \delta_{\ell+1,k}w_{\ell+1,j,k} & \text{using } \varphi_{\ell,j}(\mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\mathrm{net}_{\ell,j}) \\
\Leftarrow & \sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} = \sum_k \delta_{\ell+1,k}w_{\ell+1,j,k} & \text{using (8): } \overline{w}_{\ell+1,j,k} = \beta_{\ell+1,k}w_{\ell+1,j,k} \\
\Leftrightarrow & \sum_k \overline{\delta}_{\ell+1,k}\beta_{\ell+1,k}w_{\ell+1,j,k} = \sum_k \delta_{\ell+1,k}w_{\ell+1,j,k}, &
\end{array}$$

which holds on account of the induction hypothesis $\beta_{\ell+1,j}\overline{\delta}_{\ell+1,j} = \delta_{\ell+1,j}$.□

An induction over the number of pattern presentations, using these lemmas, concludes the proof of theorem 1. □

Variation / modification	Remarks
Momentum [Rumelhart-86]	Theorem 1 holds without modification of the network parameters.
Batch or off-line learning	Theorem 1 holds without modification of the network parameters.
Flat spot elimination [Fahlman-88]	Theorem 1 holds if the constant $\bar{c}$ added to the derivative in network N is divided by the gain ($\bar{c} = c/\beta_{\ell,j}$).
Weight discretization with multiple thresholding of the real-valued weights [Fiesler-94.2]	The thresholds need adaptation. If d and $\bar{d}$ are the discretization functions applied on the weights, theorem 1 holds if $\forall x$: $\beta_{\ell,j}\, d(x) = \bar{d}(\beta_{\ell,j}\, x)$.
Adaptive gain [Plaut-86] [Bachmann-90] [Kruschke-91] [Sperduti-93]	A change $\Delta\beta_{\ell,j}$ of the gain can be expressed as a change of the learning rates from $\beta^2_{\ell,j}\eta_{\ell,j}$ to $(\beta_{\ell,j} + \Delta\beta_{\ell,j})^2\eta_{\ell,j}$, and of the weights from $\beta_{\ell,j}\mathbf{w}_{\ell,j}$ to $(\beta_{\ell,j} + \Delta\beta_{\ell,j})\mathbf{w}_{\ell,j}$, without changing the gain of the activation functions.
Faster convergence by using steeper activation functions [Izui-90] [Cho-91]	Is equivalent to using a higher learning rate and a bigger weight range.
Increasing gain to obtain a hard limiting threshold [Corwin-94] [Yu-94]	An alternative to increasing the gain is multiplying the weights with a constant >1 and to replace in the final stage of the training process the activation function by a threshold when it does not cause a degradation in performance.

Table 2: Theorem 1 applied to variations and modifications of the backpropagation algorithm

4 Extensions and Applications of Theorem 1

Instead of the standard backpropagation algorithm, as described in section 3, variations of it are often used. Table 2 gives a short discussion of some common variations of the backpropagation algorithm. The corresponding proofs are omitted, as they are analogous to the proof of theorem 1.

A major problem in using optical activation functions is their non-standard gain [Saxena-94]. In figure 2, an optical activation function and its approximation by a shifted sigmoid, with a gain of approximately $1/24$, are depicted. Note that the domain of the optical activation function is restricted to positive values due to constraints imposed by the optical implementation. Using this optical activation function in a backpropagation algorithm with a normal learning rate, say 0.3, and a normal initial weight interval, say $[-0.5, 0.5]$, leads to very slow convergence. Theorem 1 explains why: this choice of parameters corresponds to having an activation function of gain one and a small learning rate of $(1/24)^2 \cdot 0.3 = 0.00052$. Theorem 1 gives a way to overcome this problem: choose a learning rate of $24^2 \cdot 0.3 = 172.8$ and an initial weight interval of $[-(24 \cdot 0.5), 24 \cdot 0.5]$. The neural network using these adopted parameters performed well.

5 Conclusions

The gain of the activation function and the learning rate in backpropagation neural networks are exchangeable. This also holds for several variations of the backpropagation algorithm, like the use of momentum terms, adaptive gain algorithms, and Fiesler's weight discretization method. The relationship between gain, learning rate, and weights can be used to obtain better performance whenever activation functions with a non-standard gain are used, as for example in hardware implementations of activation functions.

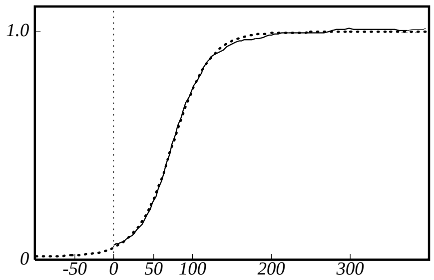

Figure 2: An optical activation function (solid line) and its approximation by a shifted sigmoid (dotted line).

References

[Bachmann-90] C. M. Bachmann. **Learning and Generalization in Neural Networks**. PhD thesis, Department of Physics, Brown University, Providence, RI, USA, May 1990.

[Cho-91] T.-H. Cho, R. W. Conners, and P. A. Araman. **Fast Back-Propagation Learning Using Steep Activation Functions and Automatic Weight Reinitialization**. *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics: Decision Aiding for Complex Systems*, volume 3, pages 1587–1592, Charlottesville, USA, October 13–16, 1991. ISBN: 0–7803–0223–8.

[Codrington-94] C. Codrington and M. Tenorio. **Adaptive Gain Networks**. *Proceedings of the IEEE International Conference on Neural Networks (ICNN94)*, volume 1, pages 339–344, Orlando, Florida, June 27–29, 1994. IEEE Catalog Number 94CH3429–8.

[Corwin-94] E. Corwin, A. Logar, and W. Oldham. **An Iterative Method for Training Multilayer Networks with Threshold Functions**. *IEEE Transactions on Neural Networks*, volume 5, number 3, pages 507–508, May 1994.

[Fahlman-88] S. E. Fahlman. **An Empirical Study of Learning Speed in Backpropagation Networks**. Technical Report CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Sep. 1988.

[Fiesler-94.1] E. Fiesler. **Neural Network Classification and Formalization**. *Computer Standards & Interfaces*, special issue on Neural Network Standardization, editor: J. Fulcher, volume 16, number 3, pages 231–239, June 1994, North-Holland / Elsevier Science Publishers, Amsterdam, The Netherlands. ISSN: 0920-5489.

[Fiesler-94.2] E. Fiesler, A. Choudry, and H. J. Caulfield. **A Universal Weight Discretization Method for Multi-Layer Neural Networks**. *IEEE Transactions on Systems, Man, and Cybernetics (IEEE-SMC)*, (Accepted for publication).
See also: 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*, volume SPIE 1281, pages 164–173, SPIE, Bellingham, Washington, 1990. ISBN: 0-8194-0328-8.

[Izui-90] Y. Izui and A. Pentland. **Speeding Up Back Propagation**. *Proceedings of the International Joint Conference on Neural Networks (IJCNN90)*, editor: M. Caudill, volume 1, pages 639–642, Washington, DC, January 15–19, 1990, Lawrence Erlbaum Associates, Hillsdale, New Jersey. ISBN: 0–8058–0754–3.

[Kruschke-91] J. K. Kruschke and J. R. Movellan. **Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Backpropagation Networks**. *IEEE Transactions on Systems, Man and Cybernetics*, volume 21, number 1, pages 273–280, January 1991.

[Mundie-92] D. B. Mundie and L. W. Massengil. **Threshold Non-Linearity Effects on Weight-Decay Tolerance in Analog Neural Networks**. *Proceedings of the International Joint Conference on Neural Networks (IJCNN92)*, volume 2, pages 583-587, Baltimore, MD, USA, June 7-11, 1992. ISBN: 0–7803–0559-0.

[Plaut-86] D. Plaut, S. Nowlan, and G. Hinton. **Experiments on Learning by Back Propagation**. Technical report CMU-CS-86-126, Carnegie-Mellon University, Pittsburgh, USA, 1986.

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

[Saxena-94] I. Saxena and E. Fiesler. **Adaptive Multilayer Optical Neural Network with Optical Thresholding.** Invited paper for *Optical Engineering*, special on Optics in Switzerland (P. Rastogi, editor), ISSN: 0091-3286, submitted.

[Sperduti-93] A. Sperduti and A. Starita. **Speed Up Learning and Network Optimization with Extended Backpropagation**. *Neural networks*, volume 6, pages 365–383, 1993, Pergamon Press.

[Thimm-94] G. Thimm and E. Fiesler. **Weight Initialization for High Order and Multilayer Perceptrons**. *IEEE Transactions on Neural Networks*, submitted.

[Wessels-92] L. F. A. Wessels and E. Barnard. **Avoiding false local minima by proper initialization of connections**. *IEEE Transactions on Neural Networks*, volume 3, number, pages 899–905, November 1992.

[Yu-94] X. Yu, N. Loh, and W. Miller. **Training Hard-Limiting Neurons Using Back-Propagation Algorithm by Updating Steepness Factors**. *Proceedings of the IEEE International Conference on Neural Networks (ICNN94)*, volume 1, pages 526–530, Orlando, Florida, June 27–29, 1994, IEEE Catalog Number 94CH3429–8.

[Zurada-92] J. M. Zurada. **Introduction to Artificial Neural Systems**. West Publishing Company, 1992. ISBN: 0–314–93391–3.