Reprint: The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks

Appeared in **Neural Computation 8(2), Feb. 1996**

Georg Thimm
Perry Moerland
Emile Fiesler
IDIAP, CH-1920 Martigny, Switzerland, Electronic mail: Thimm@IDIAP.CH

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.

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 for φ 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.

Several other authors hypothesized about the existence of a relationship between the gain of the activation function and the weights (Codrington and Tenorio 1994; Wessels and Barnard 1992)[3], or between the gain and the learning rate (Kruschke and Movellan 1991; Mundie and Massengill 1992; Zurada 1992; Brown *et al.* 1993; Brown and Harris 1994). In specific, J. Zurada writes: "*... leads to the conclusion that using activation functions with large* **[gain]** λ *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 *...*".

[1] See for example (Rumelhart *et al.* 1986) .

[2] Note that gain and steepness are identical for activation functions with $\gamma = 1$ (Saxena and Fiesler 1995). The term *temperature* is sometimes used as a synonym for the reciprocal of gain.

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

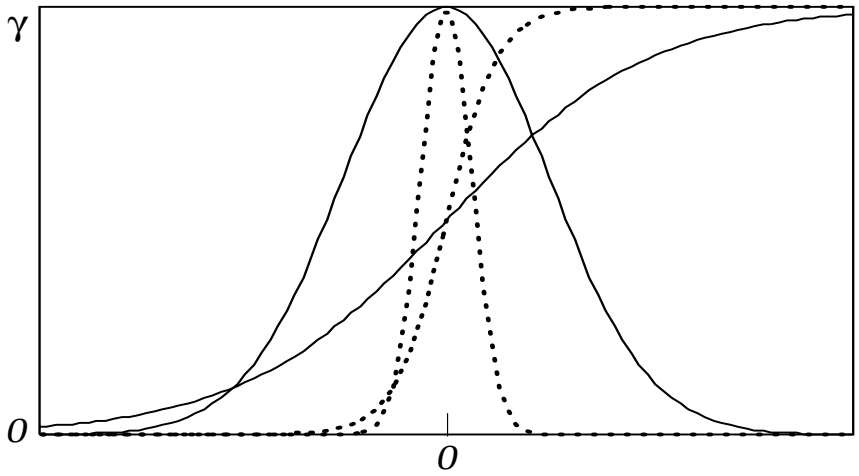

Figure 1: A logistic and a Gaussian function of gain one (solid lines) and their four times steeper counterparts (dotted lines).

	Network M	Network N
Activation function	$\varphi(x) = \overline{\varphi}(\beta x)$	$\overline{\varphi}(x)$
Gain	β	$\bar{\beta} = 1$
Learning rate	η	$\bar{\eta} = \beta^2 \eta$
Weights	$\mathbf{w}$	$\bar{\mathbf{w}} = \beta \mathbf{w}$

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

2 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 where corresponding neurons have activation functions φ_i and $\bar{\varphi}_i$ (indices are omitted where corresponding functions or variables have the same index):

$$\varphi(x) = \bar{\varphi}(\beta x). \tag{2}$$

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 β. A proof of theorem 1 and a detailed description of the backpropagation algorithm can be found in the appendix.

Theorem 1 Two neural networks M and N of identical topology whose activation function φ, gain β, learning rate η, and weights $\mathbf{w}$ are related to each other as given in table 1, are equivalent under the on-line backpropagation algorithm; that is, when presented the same pattern set in the same order, their outputs are identical.

An increase of the gain with a factor β can therefore be compensated for by dividing the initial weights by β and the learning rate by β^2.

3 Extensions and Applications of Theorem 1

Many variations of the standard backpropagation algorithm are in use. A list of common variations and their interdependence with theorem 1 is presented here. The corresponding proofs are omitted, as they are analogous to the proof of theorem 1.

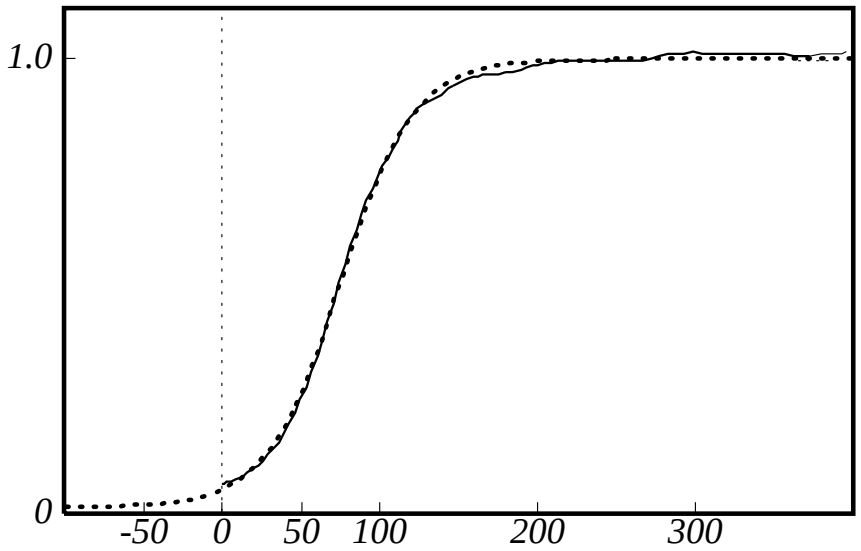

Figure 2: An optical activation function (solid line) and its approximation by a shifted sigmoid (dotted line).

Momentum (Rumelhart *et al.* 1986): Theorem 1 holds when both networks have identical momentum terms.

Batch or off-line learning: Theorem 1 holds without modification of the network parameters when off-line learning is used.

Flat spot elimination (Fahlman 1988): Theorem 1 holds if the constant $\bar{c}$ (0.1 in S. Fahlman's paper) added to the derivative in network N equals c/β.

Weight discretization with multiple thresholding of the real-valued weights (Fiesler *et al.* 1995) requires an adaptation of the threshold values for the weight discretization. If d and $\bar{d}$ are the discretization functions applied on the weights, theorem 1 holds if $\forall x : \beta d(x) = \bar{d}(\beta x)$.

Adaptive gain (Plaut *et al.* 1986; Bachmann 1990; Kruschke and Movellan 1991): A change $\Delta\beta$ of the gain can be expressed as a change of the learning rates from $\beta^2\eta$ to $(\beta + \Delta\beta)^2\eta$, and of the weights from $\beta\mathbf{w}$ to $(\beta + \Delta\beta)\mathbf{w}$, without changing the gain of the activation functions.

The use of steeper activation functions for decreasing the convergence time (Izui and Pentland 1990; Cho *et al.* 1991) is equivalent to using a higher learning rate and a bigger weight range according to theorem 1.

Approaching hard limiting thresholds by increasing the gain of the activation functions (Corwin *et al.* 1994; Yu *et al.* 1994) is similar to multipling the weights with a constant greater than one. In the final stage of the training process the activation functions can be replaced by a threshold if this does not cause a degradation in performance.

A major problem in using optical activation functions is their non-standard gain (Saxena and Fiesler 1995). 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.

4 Conclusions

The gain of the activation function and the learning rate in backpropagation neural networks are exchangeable. More precisely, there exists a well-defined relation ship between the gain, the learning rate,

and the initial weights. This relation ship is presented as a theorem that is accompanied by a detailed, yet easy to understand, proof.

The theorem also holds for several variations of the backpropagation algorithm, like the use of momentum terms, adaptive gain algorithms, and Fiesler's weight discretization method.

A direct application area of the theorem is analog neural network hardware implementations, where the possible activation functions are limited by the available components. One can compensate for their non-standard gain by modifying the learning rate and initial weight according to the theorem to optimize the performance of the neural network.

Appendix

Before proving theorem 1 a generalization of the on-line backpropagation learning rule (Rumelhart *et al.* 1986) is described, in which every neuron has its own (local) learning rate and activation function. The standard case of a unique learning rate and activation function corresponds to all local learning rates and activation functions being equal for the whole network.

The following notation and nomenclature is used (Fiesler 1994): 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 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[4].

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 layer wise through the network. The new activation value of neuron j in layer ℓ ($2 \leq \ell \leq L$) is

$$a_{\ell,j} = \begin{cases} 1 & \text{if } j = 0 \\ \varphi_{\ell,j}(\text{net}_{\ell,j}) & \text{otherwise,} \end{cases} \tag{3}$$

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

$$\text{net}_{\ell,j} = \sum_{i=0}^{N_{\ell-1}} w_{\ell,i,j} a_{\ell-1,i}, \tag{4}$$

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

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:

$$\begin{aligned} \delta_{L,j} &= \varphi'_{L,j}(\text{net}_{L,j})\ (t_j - a_{L,j}) && \text{for the output layer } L \\ \delta_{\ell,j} &= \varphi'_{\ell,j}(\text{net}_{\ell,j})\ \textstyle\sum_k \delta_{\ell+1,k} w_{\ell+1,j,k} && \text{for layers } 2 \leq \ell \leq L-1. \end{aligned} \tag{5}$$

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

$$w_{\ell,i,j} := w_{\ell,i,j} + \eta_{\ell,j} \delta_{\ell,j} a_{\ell-1,i}, \tag{6}$$

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

[4] See (Thimm and Fiesler 1995) for an in-depth study of weight initialization.

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

The notation introduced in the formulation of the backpropagation algorithm permits the proper formulation of the

Proof of Theorem 1 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, (4) 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 overlined; 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:

$$\begin{aligned}
& a_{\ell,j} = \overline{a}_{\ell,j} && \text{using (3), trivially fulfilled for } j=0. \\
\Leftrightarrow\ & \varphi_{\ell,j}(\mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\overline{\mathrm{net}}_{\ell,j}) && \text{using (2): } \varphi_{\ell,j}(x) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\, x) \\
\Leftrightarrow\ & \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\ \mathrm{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\overline{\mathrm{net}}_{\ell,j}) && \\
\Leftarrow\ & \beta_{\ell,j}\ \mathrm{net}_{\ell,j} = \overline{\mathrm{net}}_{\ell,j} && \text{using (7)} \\
\Leftrightarrow\ & \beta_{\ell,j}(\mathbf{w}_{\ell,j} \cdot \mathbf{a}_{\ell-1}) = \overline{\mathbf{w}}_{\ell,j} \cdot \overline{\mathbf{a}}_{\ell-1} && \text{on account of } \overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\mathbf{w}_{\ell,j} \\
\Leftrightarrow\ & \beta_{\ell,j}(\mathbf{w}_{\ell,j} \cdot \mathbf{a}_{\ell-1}) = (\beta_{\ell,j}\mathbf{w}_{\ell,j}) \cdot \overline{\mathbf{a}}_{\ell-1}, &&
\end{aligned}$$

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:

$$\begin{aligned}
& \Delta\overline{\mathbf{w}}_{\ell,j} = \beta_{\ell,j}\Delta\mathbf{w}_{\ell,j} && \text{definition of } \Delta\mathbf{w}_{\ell,j} \\
\Leftrightarrow\ & \overline{\eta}_{\ell,j}\overline{\delta}_{\ell,j}\overline{\mathbf{a}}_{\ell-1} = \beta_{\ell,j}\eta_{\ell,j}\delta_{\ell,j}\mathbf{a}_{\ell-1} && \text{since } \overline{\eta}_{\ell,j} = \beta_{\ell,j}^2\eta_{\ell,j} \text{ and } \overline{\mathbf{a}}_\ell = \mathbf{a}_\ell \\
\Leftarrow\ & \beta_{\ell,j}^2\eta_{\ell,j}\overline{\delta}_{\ell,j} = \beta_{\ell,j}\eta_{\ell,j}\delta_{\ell,j}, &&
\end{aligned}$$

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:

$$
\begin{aligned}
& \beta_{L,j}\overline{\delta}_{L,j} = \delta_{L,j} && \text{using (5)} \\
\Leftrightarrow\ & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\overline{\text{net}}_{L,j})\ (t_j - \overline{a}_{L,j}) & \\
& \quad = \varphi'_{L,j}(\text{net}_{L,j})\ (t_j - a_{L,j}) && \text{since } \overline{a}_{L,j} = a_{L,j} \\
\Leftarrow\ & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\overline{\text{net}}_{L,j}) = \varphi'_{L,j}(\text{net}_{L,j}) && \text{using } \overline{\text{net}}_{L,j} = \beta_{L,j}\,\text{net}_{L,j} \\
\Leftrightarrow\ & \beta_{L,j}\ \overline{\varphi}'_{L,j}(\beta_{L,j}\,\text{net}_{L,j}) = \varphi'_{L,j}(\text{net}_{L,j}), &
\end{aligned}
$$

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

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

$$
\begin{aligned}
& \beta_{\ell,j}\overline{\delta}_{\ell,j} = \delta_{\ell,j} && \text{using (5)} \\
\Leftrightarrow\ & \beta_{\ell,j}\ \overline{\varphi}'_{\ell,j}(\overline{\text{net}}_{\ell,j})\ \textstyle\sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} & \\
& = \varphi'_{\ell,j}(\text{net}_{\ell,j})\ \textstyle\sum_k \delta_{\ell+1,k} w_{\ell+1,j,k} && \text{using } \overline{\text{net}}_{\ell,j} = \beta_{\ell,j}\,\text{net}_{\ell,j} \\
\Leftrightarrow\ & \beta_{\ell,j}\ \overline{\varphi}'_{\ell,j}(\beta_{\ell,j}\,\text{net}_{\ell,j})\ \textstyle\sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} & \\
& = \varphi'_{\ell,j}(\text{net}_{\ell,j})\ \textstyle\sum_k \delta_{\ell+1,k} w_{\ell+1,j,k} && \text{using } \varphi_{\ell,j}(\text{net}_{\ell,j}) = \overline{\varphi}_{\ell,j}(\beta_{\ell,j}\,\text{net}_{\ell,j}) \\
\Leftarrow\ & \textstyle\sum_k \overline{\delta}_{\ell+1,k}\overline{w}_{\ell+1,j,k} = \sum_k \delta_{\ell+1,k} w_{\ell+1,j,k} && \text{using (8): } \overline{w}_{\ell+1,j,k} = \beta_{\ell+1,k} w_{\ell+1,j,k} \\
\Leftrightarrow\ & \textstyle\sum_k \overline{\delta}_{\ell+1,k}\beta_{\ell+1,k} w_{\ell+1,j,k} = \sum_k \delta_{\ell+1,k} w_{\ell+1,j,k}, &
\end{aligned}
$$

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. □

Addendum

For completeness the authors would like to include the reference to a letter in Neural Networks (Jia *et al.* 1994), that was brought to their attention after the submission of this paper to Neural Computation, in which a similar theorem is presented, albeit without proof or applications. The theorem includes momentum and is related to (Izui and Pentland 1990) and (Sperduti and Starita 1993) by the authors.

References

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

Brown, M., An, P.C., Harris, C.J., and Wang H. 1993. How Biased is your Multi-Layer Perceptron. *World Congress on Neural Networks*, **3**, 507–511, Portland, Oregon, USA.

Brown, M. and Harris, C. 1994. Neurofuzzy Adaptive Modelling and Control. *Prentice Hall International Series in Systems and Control Engineering*, M. J. Grimble, ed., Prentice Hall. ISBN: 0–13–134453–6.

Cho, T.-H., Conners, R. W., and Araman, P. A. 1991. 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*, **3**, 1587–1592, Charlottesville, USA. ISBN: 0–7803–0223–8.

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

Corwin, E., Logar, A., and Oldham, W. 1994. An Iterative Method for Training Multilayer Networks with Threshold Functions. *IEEE Transactions on Neural Networks*, **5**(3), 507–508.

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

Fiesler, E. 1994. Neural Network Classification and Formalization. *Computer Standards & Interfaces*, special issue on Neural Network Standardization, Fulcher, J., ed., **16**(3), 231–239, North-Holland / Elsevier Science Publishers, Amsterdam, The Netherlands. ISSN: 0920-5489.

Fiesler, E., Choudry, A., and Caulfield, H. J. 1995. A Universal Weight Discretization Method for Multi-Layer Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics (IEEE-SMC)*, accepted for publication.

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

Izui, Y., and Pentland, A. 1990. Analysis of Neural Networks With Redundancy. *Neural Computation*, **2**, 226-238.

Jia, Q., Hagiwara, K., Toda, N. and Usui, S. 1994. Equivalence Relation Between the Backpropagation Learning Process of an FNN and That of an FNNG, *Neural Networks*, **7**(2), 411, Pergamon Press.

Kruschke, J. K., and Movellan, J. R. 1991. Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Backpropagation Networks. *IEEE Transactions on Systems, Man and Cybernetics*, **21**(1), 273–280.

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

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

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

Saxena, I., and Fiesler, E. 1995. Adaptive Multilayer Optical Neural Network with Optical Thresholding. Invited paper for *Optical Engineering*, special on Optics in Switzerland, Rastogi, P., ed. ISSN: 0091–3286, in press.

Sperduti, A., and Starita, A. 1993. Speed Up Learning and Network Optimization with Extended Backpropagation. *Neural Networks*, **6**, 365–383, Pergamon Press.

Thimm, G., and Fiesler, E. 1995. Weight Initialization for High Order and Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, submitted.

See also: Thimm, G., and Fiesler, E. 1994. Weight Initialization for High Order and Multilayer Perceptrons. *Proceedings of the '94 SIPAR–Workshop on Parallel and Distributed Computing*, Aguilar, M., ed., 91–94, Institute of Informatics, University Pérolles, Chemin du Musée 3, Fribourg, Switzerland. SI Group for Parallel Systems.

Wessels, L. F. A., and Barnard, E. 1992. Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks*, **3**, 899–905.

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

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