Discrete All-Positive Multilayer Perceptrons for Optical Implementation

P. Moerland [a] E. Fiesler [b] I. Saxena [b]

IDIAP–RR 97-02

February 1997

revised in November 1997

published in
Optical Engineering, vol. 37, no. 4, April 1998

[a] IDIAP, CP 592, CH-1920 Martigny, Switzerland, E-mail: Perry.Moerland@idiap.ch
[b] Physical Optics Corporation, Gramercy Place, Torrance, CA 90501, USA

IDIAP Research Report 97-02

Discrete All-Positive Multilayer Perceptrons for Optical Implementation

P. Moerland E. Fiesler I. Saxena

February 1997

revised in November 1997

published in
Optical Engineering, vol. 37, no. 4, April 1998

Abstract. All-optical multilayer perceptrons differ in various ways from the ideal neural network model. Examples are the use of non-ideal activation functions which are truncated, asymmetric, and have a non-standard gain, restriction of the network parameters to non-negative values, and the limited accuracy of the weights. In this paper, a backpropagation-based learning rule is presented that compensates for these non-idealities and enables the implementation of all-optical multilayer perceptrons where learning occurs under control of a computer. The good performance of this learning rule, even when using a small number of weight levels, is illustrated by a series of computer simulations incorporating the non-idealities.

1 Introduction

An important feature of multilayer perceptrons (MLPs) is their massive parallelism of weighted interconnections between layers of non-linear processing elements. Conventional digital computers, however, cannot take advantage of the parallelism inherent in MLP computation. A promising alternative is the use of optics which offers the potential of parallel three-dimensional interconnections in a compact way, using, for example, spatial light modulators as two-dimensional weighting devices [1]. In addition to matrix-vector multiplication that is performed optically, efficient optical MLP (OMLP) implementations should incorporate a successful optical thresholding technique for non-linear processing to preserve the parallelism inherent in optics, in contrast to the use of electronic non-linear thresholding [2, 3, 4]. Such a technique avoids photo-electric conversion of light and eliminates electronic thresholding. An adaptive OMLP trained under computer control which could implement this was described earlier [5, 6]. This architecture consists of liquid crystal television screens to implement the inputs and the weights [7], and liquid crystal light valves (LCLVs) to implement non-linear thresholding.

The inclusion of commercially available LCLVs to implement optical thresholding into the backpropagation rule for MLPs has been reported in a previous paper by the authors [8]. In specific, this paper [8] has resolved the constraints of non-standard non-linear thresholding and its adapted backpropagation rule compensates for the gain (steepness) of the activation function by modification of the initial values of training parameters, showing good performance in computer simulations including laboratory data for the LCLVs.

Following up the work on the adapted algorithm [8], in this paper our method further resolves the restriction on the weights to non-negative values and to a small number of discrete levels, while integrating the use of LCLVs for optical thresholding. All-positive network parameters stem from the intensity-modulation based OMLP processing scheme. The use of discrete weights is required for hardware implementations in electronics for chip area minimization and in analog electronics and optics for improved discrimination of analog quantities. These aspects are discussed in more detail in section 2.

The structure of the article is as follows. In Section 2, the use of LCLV response curves for optical thresholding is explained and it is shown how to compensate for their translation along the x-axis and their non-standard gain. Section 3 starts with a discussion and description of the transformation of the network weights to non-negative values to compensate for the lack of optical subtraction. Then, the need of quantized weight values and a weight discretization method that is applicable to OMLPs are described. Next, the benchmarks and parameters employed in the computer simulations are outlined. A first series of simulations evaluates the use of LCLV response curves and subtraction compensation for non-negative OMLPs trained with the (adapted) backpropagation algorithm. A range of computer simulations with the LCLV response data demonstrates the suitability of the adapted weight discretization method for non-negative MLPs. Finally, some first results of recall on an optical perceptron for handwritten digit recognition are presented.

2 LCLV Activation Functions

Foremost amongst optical thresholding devices are liquid crystal light valves with their non-linear, sigmoid-like optical activation functions [9, 10], in addition to their practical characteristics (low operating voltage, high contrast ratio at visible wavelengths, and low intensities required) which make them easy-to-use devices.

The response data of the LCLV, or the optical activation functions, consist of a set of non-negative x- and y-coordinates representing the write light irradiance and the read-out light irradiance of the LCLV. A comparison of properties (translation, asymptotes, and steepness) of five sigmoid-like responses of four different LCLVs was made earlier [5] based on close approximations [10, 11], with a generic sigmoid curve fit.

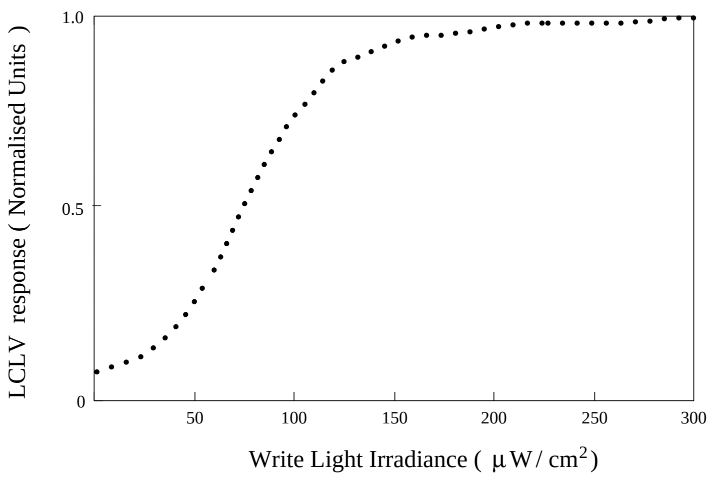

Figure 1: Response curve of a LCLV device.

	LCLV1	LCLV2	LCLV3	LCLV4a	LCLV4b
β	0.087	0.0062	0.043	0.79	1.40
Midpoint	18.0	146.0	74.0	1.00	3.30
Tangent	0.03	0.003	0.011	0.32	0.263
β_{new}	0.12	0.012	0.044	1.28	1.052

Table 1: Estimation of the gain based on the generic curve fit (β) and on measured LCLV response data (β_{new}).

The main differences between the LCLV response curves and the standard sigmoid, $1/(1+e^{-x})$, are that they are located in the non-negative quadrant and have a non-standard gain (steepness) (Figure 1 gives a typical example). The backpropagation algorithm has been adapted [8] to compensate for these differences; this solution is summarized in section 2.1.

To perform computer simulations with the measured non-negative LCLV response data, a continuous approximation by linear interpolation is used here. The derivative of the response data curve, which is needed in the backward pass of the backpropagation algorithm, is defined to be the derivative of this linear interpolation. The *midpoint* of the interpolated data curves is defined as the x-value corresponding to a normalized y-value of a half, in analogy with the standard sigmoid.

2.1 Adaptations for the LCLV Activation Functions

Simulation results [8] have shown that the backpropagation algorithm [12] with a standard choice for the initial parameters (typically weights in a small interval symmetric around zero and a learning rate less than 1) fails to converge when using the LCLV response data as non-linear thresholding functions. In this section, an adapted backpropagation learning rule is described that consists in modifying the initial training conditions as well as compensating for the gain of the activation function.

The initial weights for a multilayer perceptron are best chosen uniformly distributed in an interval symmetric around zero [13]. However, this initialization method leads to non-convergence results when using a sigmoidal activation function which has been translated along the x-axis, like the LCLV response curves. Therefore, a weight initialization method resulting in neuron inputs centered around the midpoint of the activation function has been used instead [8].

The curve fitting of the five LCLV response curves with a generic sigmoid translated along the

	Network M	Network N
Activation function	$\varphi(x)$	$\varphi(\beta x)$
Gain	1	β
Learning rate	η	η/β^2
Weights	$\boldsymbol{w}$	$\boldsymbol{w}/\beta$

Table 2: The relationship between activation function, gain, weights, and learning rate when increasing the gain with a factor β.

x-axis by δ and along the y-axis by α, a range γ, and a gain β:

$$\phi(x) = \alpha + \left(\frac{\gamma}{1 + e^{-\beta x + \delta}}\right),$$

showed that the gain parameters β of LCLV1-3 differ greatly from the standard value of one [5] (see the first row of Table 1). Such a non-standard gain cannot be combined with rules of thumb for choosing the parameters of the backpropagation learning rule and does not guarantee convergence of network training. This influence can be eliminated by applying a simple and precise relationship (Table 2) that enables compensating for the non-standard gain in backpropagation neural networks by changing the learning rate and the initial weights [15] while maintaining equivalent network behaviour during training. The change of the gain with a factor β (as in network N in Table 2) can therefore be compensated for by dividing the initial weights by β and the learning rate by β^2.

In this paper, the respective gains of the five LCLV response curves are directly calculated from their sampled data and not based on the generic curve fits as in our previous work [8]. This is founded on the analogy between the LCLV response curves and a sigmoidal function that has been translated along the x-axis:

$$\phi(x) = \frac{1}{1 + e^{-\beta x + \delta}}.$$

This function has the property that its gain β is equal to 4 times the tangent in its midpoint (which is also its inflection point). This relationship can easily be applied to the measured LCLV response data by determining their midpoints, estimating the tangents at these midpoints, and multiplying these by 4 to get an estimate of the gain. The resulting values are presented in Table 1 and show once more the non-standard gains β_{new} of the LCLV responses. The advantage of these new gain estimates is that they are not biased towards a decidedly negative lower asymptote, as is the case with the gain estimates obtained from the generic sigmoid. It should also be noted that the difference between β and β_{new} is quite small (less than a factor of 2), so that actual difference in performance as compared with those in Ref. 6 is expected to be small.

3 Discrete Non-Negative Multilayer Perceptrons: Theory and Experiments

In this section, the subtraction compensation method is described and how it can be combined with an adaptation of the weight discretization method [16]. The performance of these techniques is evaluated in a series of experiments on training all-positive discrete MLPs with five different LCLV response curves as non-linear activation functions.

3.1 On Subtraction Compensation and Weight Discretization

3.1.1 Subtraction Compensation

In optical neural networks where information is coded in light intensity, as in our ONN [5], all variables including the interconnection weights can only be represented by non-negative quantities. This lack of negative values and the intricacy of an optical mechanism for irradiance subtraction are important limiting factors for the development of optical neural networks. Optical subtraction based on superposing polarized light intensities has only been demonstrated in rather simple optical systems [14], and it is practically impossible to realize in massively parallel optical neural networks. Some other options are offered in the use of different encoding schemes such as phase or wavelength encoding, but these techniques are virtually unexplored. Subtraction is, therefore, usually realized as an electronic difference of two photo-detected quantities that have been separated spatially [17], temporally [2], or by polarization encoding [18]. Such schemes prevent all-optical neural processing at hidden layers and necessitate serial processing for electronic thresholding instead of spatial parallelism using LCLVs. In this publication, a method is used that is based on a mathematical technique for subtraction compensation, which offers a scheme for implementing all-positive neural networks that are trained under computer control as described earlier [19]. This solution is based on a transformation of the network weights to the positive domain, enabling uninterrupted forward propagation of light in OMLPs.

The untransformed weight from neuron i to neuron j is denoted by w_{ij}, the untransformed bias of neuron j is denoted by θ_j, and its activation value is indicated as a_j (see the Appendix for a detailed description of MLPs). Transformed all-positive weights w''_{ij} have been obtained [19]:

$$w''_{ij} = max\{w'_{ij}(1 - \frac{\theta'_j}{\sum_i w'_{ij} a_i}), 0\}, \tag{1}$$

where

$$w'_{ij} = w_{ij} - w_{min},$$

and

$$\theta'_j = \theta_j - w_{min} \sum_i a_i,$$

where w_{min} is the minimum over all original weight values w_{ij} and bias values θ_j. The all-positive argument of the non-linear function ϕ is:

$$\sum_i w_{ij} a_i - \theta_j = \sum_i w''_{ij} a_i.$$

The final all-positive weights w''_{ij} can therefore easily be obtained by carrying out the transformations on a host computer and can then be mapped on a weight implementation device for matrix-vector multiplication of the transformed weight matrix w''_{ij} and activation values a_i, like a liquid crystal television (LCTV).

An example of the resulting non-negative networks where the LCLV4b data curve is used as sigmoidal activation function is shown in Figure 2. This gives a solution of the XOR problem ($xor(x, y) = xy + \overline{xy}$) with only two hidden neurons and two non-negative weight levels: 0 and 6.985. Four different networks are depicted corresponding to the fact that the non-negative weights, w''_{ij}, depend on the (four) input patterns (activation values a_i in Eq. (1)).

3.1.2 Weight Discretization

It is required to have discrete weights in electronic and optical neural network implementations, when device operation is quantized and/or the quantization of network parameters is beneficial for reducing VLSI surface area. Examples are optically controlled photoconductive synapses [20], the strength of

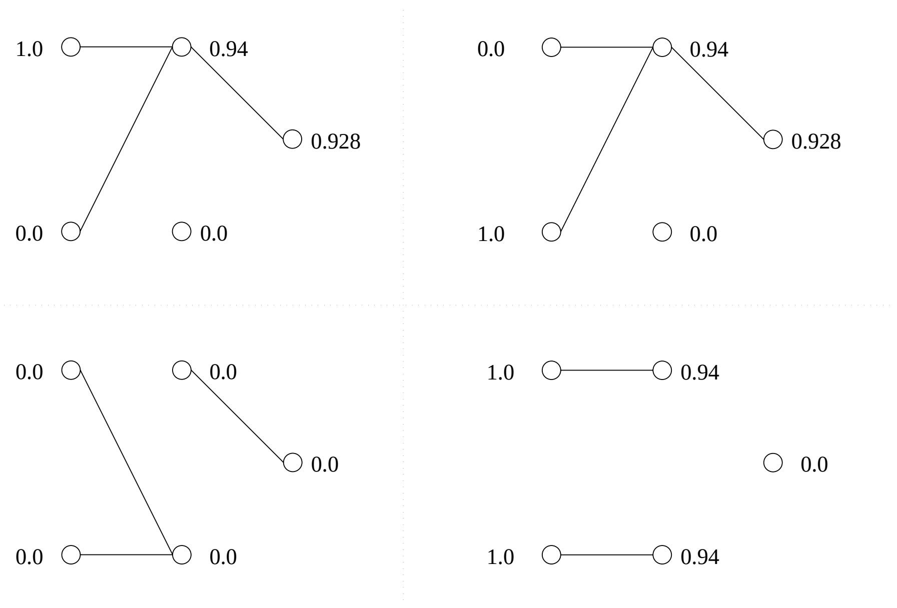

Figure 2: A solution for the XOR(2) problem with two weight levels: the absence of a connection corresponds to a zero weight otherwise the weight is equal to 6.985. Each network is depicted from left (inputs) to right (outputs).

which can be modulated by changing the discrete pixel lengths of bars of light, and a polarization-based implementation that represents each weight by 16 spatially multiplexed binary valued pixels in a ferroelectric liquid crystal (hence allowing 16 gray level weights) [21].

In the ONN architecture [5], the interconnection weights are represented by the pixels of a liquid crystal television screen that, in principle, provides 256 grey levels. However, a further reduction of the number of different weight values is beneficial for various reasons. Firstly, the mapping of weight values to an optical device is often non-linear [4, 21]. When only a few weight levels are used, it is simpler to linearize such a weight mapping and reduce the inaccuracy. Secondly, a small number of weight levels provides the possibility of using ferroelectric liquid crystals with binary valued pixels to represent the weights in an efficient way. Important additional advantages of such a device are increased processing speed, a linear mapping of the weights, and compactness [21].

To obtain successful network learning in the presence of discrete weights, an existing weight discretization method based on backpropagation [16] (described in detail in the Appendix) has been adapted for a OMLP with optical thresholding. It is easily implemented, and is suited for chip-in-the-loop learning. It integrates well with the subtraction compensation technique and can handle a precision of as low as a few bits. The basic idea of our weight discretization method is to use discrete weights in the forward pass and continuous weights in the backward pass. The continuous weights are discretized by mapping them to the closest discretization level with a multiple thresholding function.

To combine discretization of the weights with the subtraction compensation scheme for non-negative networks, a new multiple thresholding function has been designed for the discretization method described in the Appendix. The discretization levels are chosen to be equidistant, the minimal discrete weight value is zero, and the discretization levels are based on the maximal value w_{max} of the (non-negative) weights of the pre-trained continuous network. The size of the discretization step can be controlled by changing the parameter $Discr$ and the set of d discretization levels is:

$$\left\{ \frac{(n-1)\cdot(w_{max})}{(d-1)\cdot(Discr)} \quad | \quad n = 1, 2, ..., d \right\}. \tag{2}$$

Therefore, the maximal discrete weight value is equal to $w_{max}/Discr$. The combination with the transformations described in section 3.1.1 is straightforward: each w''_{ij} is mapped to the closest discretization level using the multiple thresholding function (2).

3.2 Experiments

A set of benchmark problems including three real-world problems and the eXclusive OR (XOR) problem has been used to evaluate with computer simulations the proposed techniques for training discrete, non-negative MLPs. These benchmarks have been chosen because they are well-known in the machine learning community and, in the case of the Digit benchmark, because of its practical relevance as a test bed for an optical implementation.

eXclusive OR (XOR) The training set consists of the boolean exclusive OR function. It is the classical example of a simple problem that is not linearly separable [12].

Sonar This data set was originally used by R. Gorman and T. Sejnowski in their study of the classification of sonar signals using a neural network. The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock [23]. Each pattern is a set of 60 numbers in the range $[0, 1]$. The corresponding output patterns are the two unit vectors.

Wine is the result of a chemical analysis of wines grown in a region in Italy which are derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. A wine has to be classified using these 13 values, which have been scaled to the interval $[0, 1]$. The target patterns are the three unit vectors [24].

Digit This benchmark consists of a subset of 1000 patterns out of a database of more than 20000 digitized handwritten characters [25]. Each digit was scaled to fit into a 32×32 matrix, and each pixel is represented by an eight bit value. To save computation time and space the 32×32 matrix has been converted to an 8×8 matrix, by taking the average of 4×4 sub-matrices, and scaling the input values to the interval $[0, 1]$. The target patterns are the ten unit vectors.

The benchmark characteristics and the training parameters are listed in Table 3. Two benchmarks (XOR and Sonar) have been used to evaluate straightforward on-line (weights are updated after each pattern presentation) and batch (weights are updated after presentation of the whole training set (*epoch*)) backpropagation training. In this case the criterion for training convergence was that for all patterns in the training set the absolute difference between each of the actual network outputs and their corresponding target values is at most ε_c. Actually, for the Sonar benchmark an ε_c has been used when training the continuous network and a less restrictive ε_d when training the discrete network. While the results on these benchmarks indicate the capability to learn and recall a mapping, the other two benchmarks (Wine and Digit) are used to assess the *generalization* performance, that is, the ability of the network to correctly classify examples that were not used during training. In this case, the pattern set is split up in a training, validation, and test set as is explained in more detail in section 3.2.1.

All the simulation results are averaged over a number of runs ("#Runs" in Table 3) with a different random weight initialization from the initial weight range. Each of the computer simulations is performed with the five interpolated curves of LCLV response data described in section 2. In fact, the ε criterion has been slightly refined to take into account that the minimal y-value of the LCLV response curves differs from zero (with a maximum y-intercept of 0.06 for LCLV3). Therefore, all zero-valued targets in the benchmark sets have been replaced by these minimal values to be able to compare the different curves in a fair way.

benchmark	network topology[1]	pattern set sizes			#Runs	ε_c	ε_d	learning rate	momentum	initial weight range
		train.	valid.	test						
XOR	2-2-1	4	-	-	100	0.1	0.1	0.3	0.9	$[-1,+1]$
Sonar	60-8-2	104	-	-	5	0.3	0.4	0.1	0.9	$[-0.5,+0.5]$
Wine	13-6-3	89	44	45	10	-	-	0.3	0.9	$[-0.5,+0.5]$
Digit	64-64-10	500	250	250	5	-	-	0.1	0.5	$[-0.5,+0.5]$

Table 3: Summary of the benchmarks and parameters used in the experiments.

In all simulations two standard training acceleration parameters have been used, namely a momentum term μ [12] (see Table 3) and a flat spot constant [26] of 0.1. A momentum term incorporates previous weight updates in the weight update equation (step 4 in the backpropagation algorithm described in the Appendix) to smooth out oscillations. The flat spot constant is a constant added to the derivatives of the activation function ϕ'_j (see also step 4) to prevent it from being too close to zero which would lead to very small weight updates and slow training progress. It has been shown that the gain theorem can easily be extended to include these parameters [15].

3.2.1 Measuring Generalization Performance

To validate the quality of a network during training and to test the performance afterwards, the complete available pattern set was partitioned, respecting an equal distribution of the different classes, into three sets: a *training* set with 50% of the total patterns and a *validation* set, and a *test* set, each of these with 25% of the patterns (see Table 3 for the sizes of these sets for the Wine and Digit benchmarks). The evaluation of the generalization performance is done by *cross validation* with *early stopping*, where the decision to stop training is based on the error on the validation set. This is a good way to avoid over-fitting of the network to the particular training set used. For the experiments in this section, a description of cross validation with early stopping has been used as a basis [22]. The network performance is indicated by the *mean squared error percentage* that is normalized for the number of output neurons and the number of patterns:

$$E = 100 \cdot \frac{1}{N \cdot P} \sum_{p=1}^{P} \sum_{n=1}^{N} (o_{pn} - t_{pn})^2,$$

where N is the number of output neurons of the network, P is the number of patterns in the data set considered, o_{pn} is the activation value of output neuron n for pattern p, and t_{pn} its target value for the problem at hand.

The network is trained using only the training set and after every five epochs the validation set is presented to measure the generalization error. Two different measures have been used to decide when to stop training. The first one is the *training progress* in a training strip of five epochs which gives a measure of the change in the training error over five epochs. The second one is the *generalization loss* which indicates the relative increase of the current validation error with respect to its minimal value so far. These two parameters are measured every five epochs, after presenting the validation set. Training is stopped when one of two different situations occurred: training progress sank below 0.1 (measured in parts per thousand) or the generalization loss went beyond a threshold of 5.0 (in percent) in ten consecutive measurements. This means that training is considered concluded either when the training error stagnated or when a reduction on that error lead to a successive deterioration in the generalization properties of the network. After stopping the training, the test set is presented using the network weights that provided the lowest validation error. The results presented for the benchmarks using generalization are all based on the values measured for that optimal network.

[1]Number of neurons in the input-hidden-output layers.

LCLV	Method	%Conv	# of epochs
LCLV1	On-line BP	99.0	142.9
	Batch BP	89.0	35.8
LCLV2	On-line BP	90.0	142.9
	Batch BP	83.0	37.1
LCLV3	On-line BP	99.0	109.4
	Batch BP	82.0	44.8
LCLV4a	On-line BP	74.0	116.3
	Batch BP	68.0	33.6
LCLV4b	On-line BP	100.0	132.2
	Batch BP	78.0	112.8

Table 4: Results from the simulations (average over 100 runs) with XOR(2).

LCLV	Method	%Conv	# of epochs
LCLV1	On-line BP	100.0	261.6
	Batch BP	100.0	444.8
LCLV2	On-line BP	100.0	400.0
	Batch BP	100.0	357.2
LCLV3	On-line BP	100.0	286.0
	Batch BP	100.0	407.6
LCLV4a	On-line BP	100.0	955.6
	Batch BP	100.0	411.2
LCLV4b	On-line BP	100.0	412.6
	Batch BP	100.0	499.8

Table 5: Results from the simulations (average over 5 runs) with Sonar(8).

LCLV	Method	# of epochs	Square Error Percentage			Percentage of Misclassification		
			Train.	Valid.	Test	Train.	Valid.	Test
LCLV1	On-line BP	52.5	0.47	0.03	2.58	1.01	0.00	4.77
	Batch BP	47.0	0.10	0.35	1.83	0.11	0.00	3.18
LCLV2	On-line BP	52.0	0.49	0.04	2.93	0.90	0.00	5.45
	Batch BP	51.5	0.03	0.25	1.85	0.00	0.00	3.41
LCLV3	On-line BP	80.5	0.00	0.13	1.58	0.00	0.00	2.73
	Batch BP	84.0	0.04	0.18	1.72	0.00	0.00	2.73
LCLV4a	On-line BP	93.5	0.12	0.10	2.71	0.22	0.00	4.77
	Batch BP	87.5	0.01	0.03	2.10	0.00	0.00	3.86
LCLV4b	On-line BP	57.5	0.03	0.01	2.21	0.00	0.00	3.41
	Batch BP	50.5	0.24	0.52	2.09	0.34	0.00	3.41

Table 6: Results from the simulations (average over 10 runs) with Wine(6).

In addition, results on the classification performance of the networks will also be presented. A pattern is considered correctly classified whenever the network output corresponding to the correct one is higher in value than all the other outputs. This procedure is called *winner-takes-all* (WTA) [22] and can only be used in problems whose output is implemented as a 1-of-N_L representation.

3.2.2 Experimental Results for Training Non-Negative MLPs

The experiments described in this section have been performed to evaluate the combined use of the measured LCLV response data as non-linearities and the transformation of the weights to the all-positive domain. This means that in each forward propagation step, bipolar weights w_{ij} and biases θ_j are replaced by the all-positive w''_{ij} according to Eq. (1). Furthermore, the non-standard gain, β_{new} in Table 1, of the LCLV response data is compensated for by changing the initial weight range and the learning rate as described in section 2.1: the initial weights are divided by β_{new} and the learning rate by β^2_{new}. These results with continuous (floating point) weights also give a basis for evaluating the results in training networks with discretized weights (section 3.2.3).

The results of the series of experiments with a continuous network are in Tables 4 to 7, where "%Conv" stands for the percentage of converged runs and the mean values are calculated based on the converged runs only.

An important observation is that the results for the 5 different LCLV response curves are quite similar. The only exceptions are the relatively low convergence rate for the XOR(2) benchmark with LCLV4a and the high percentage of misclassification for the Digit(64) benchmark with LCLV3. For the LCLV4a response curve the reason might be its asymmetry (with an almost absent lower asymptote) which perturbs training; for the LCLV3 response curve it might be caused by the fact that its y-intercept is quite big (around 0.06) so that network outputs are never clearly "off". This

LCLV	Method	# of epochs	Square Error Percentage			Percentage of Misclassification		
			Train.	Valid.	Test	Train.	Valid.	Test
LCLV1	On-line BP	17.0	0.70	1.80	1.29	3.00	10.56	6.32
	Batch BP	167.0	0.66	1.69	1.19	2.32	9.68	5.60
LCLV2	On-line BP	21.0	0.46	1.68	1.11	1.40	9.68	6.56
	Batch BP	190.0	0.42	1.63	1.08	1.16	9.20	6.24
LCLV3	On-line BP	10.0	1.60	2.33	1.95	8.88	13.92	10.24
	Batch BP	173.0	1.65	2.37	1.97	7.72	12.80	8.48
LCLV4a	On-line BP	90.0	0.03	1.71	1.04	0.04	10.48	6.16
	Batch BP	176.0	0.55	2.14	1.61	1.04	11.04	6.40
LCLV4b	On-line BP	32.0	0.31	1.76	1.23	0.44	10.08	6.24
	Batch BP	222.0	0.38	1.78	1.28	0.52	10.08	6.72

Table 7: Results from the simulations (average over 5 runs) with Digit(64).

LCLV	Method	%Conv	# of epochs
LCLV1	On-line BP	62.0	50.0
	Batch BP	80.0	8.3
LCLV2	On-line BP	46.0	122.7
	Batch BP	72.0	15.7
LCLV3	On-line BP	79.0	37.3
	Batch BP	73.0	144.6
LCLV4a	On-line B	71.0	177.0
	Batch BP	67.0	31.8
LCLV4b	On-line BP	68.0	383.9
	Batch BP	49.0	144.8

Table 8: Discretization results from the simulations (average over 100 runs) with XOR(2): two weight levels.

LCLV	Method	Percentage of Test Misclassification					
		Weight Discretization Levels					
		C	2	4	6	8	16
LCLV1	On-line BP	4.77	15.68	6.82	**5.91**	7.27	**6.59**
	Batch BP	3.18	37.05	**5.00**	**4.77**	**4.09**	**5.00**
LCLV2	On-line BP	5.45	9.77	**5.45**	**5.91**	**5.45**	**5.91**
	Batch BP	3.41	31.36	6.14	**4.77**	**4.77**	**5.23**
LCLV3	On-line BP	2.73	63.18	31.14	14.32	7.50	**3.86**
	Batch BP	2.73	60.00	50.45	19.32	9.55	**4.55**
LCLV4a	On-line BP	4.77	32.50	19.55	**6.14**	**3.64**	**3.41**
	Batch BP	3.86	28.18	**5.45**	**4.77**	**4.77**	**5.00**
LCLV4b	On-line BP	3.41	24.09	10.23	**3.86**	**3.41**	**3.64**
	Batch BP	3.41	61.82	22.73	12.27	8.64	**4.77**

Table 9: Discretization results from the simulations (average over 10 runs) with Wine(6).

illustrates that, in general, our adapted learning rule compensates successfully for the characteristics of the LCLV response curves. Also, the difference between the use of on-line or batch learning is quite small. On-line backpropagation shows the best results in terms of percentage of converged runs for the XOR(2) benchmark and in terms of number of iterations for Digit(64). Batch learning, on the other hand, shows fast convergence on XOR(2) and gives the best classification rate on the test set for the Wine problem.

A comparison of these results with the ones obtained in a benchmarking study (also including Xor(2), Sonar, Wine, and Digit benchmarks) of different adaptive learning rate algorithms for ordinary MLPs [27], shows that the influence of the optical non-idealities is as good as negligible and gives comparable results.

3.2.3 Experimental Results for Training Discrete Non-Negative MLPs

For most benchmarks, the number of discretization levels, d, used in the simulations is subsequently 2, 4, 6, 8, and 16. For the more difficult Digit benchmark, d has been chosen equal to 2, 4, 8, 16, 32, or 64. The networks resulting from continuous pre-training in section 3.2.2 have been used as initial networks. Based on the results of some initial simulations the parameter *Discr* (Eq. (2)) was chosen equal to one for the XOR(2) benchmark and equal to two for all other benchmarks. The number of epochs included in Tables 8 to 12 is the number of epochs used for the discrete training only, and does not include the continuous pre-training.

In Table 8, an overview of the discretization results for the XOR(2) benchmark with as little as two weight levels is given. When comparing these with the results for continuous pre-training

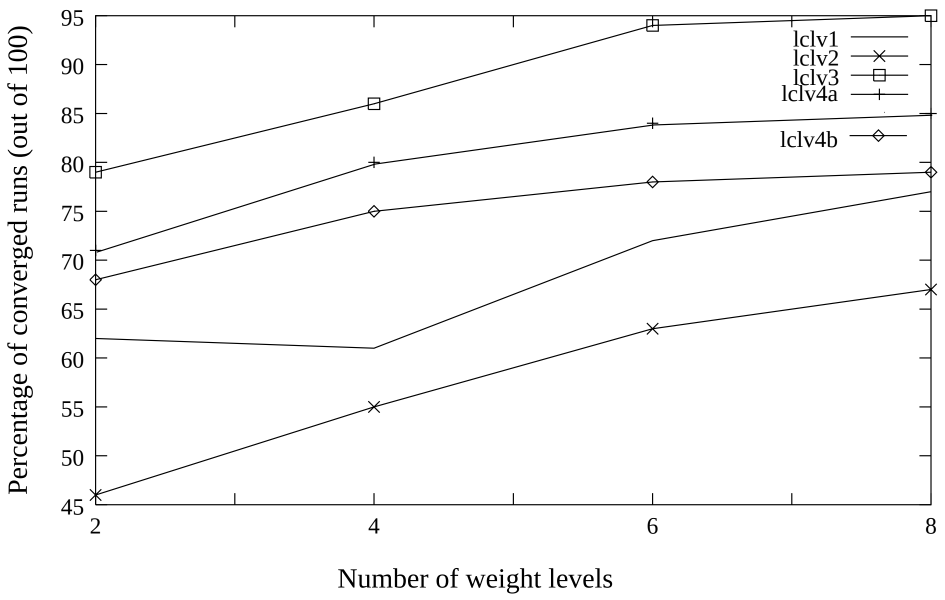

Figure 3: The number of weight levels versus the percentage of converged experiments for XOR(2) with on-line backpropagation.

(Table 4), one observes that the percentage of converged runs for discrete training is considerably lower (most notably for LCLV2 and on-line training where the percentage is halved). The evolution of the percentage of converged runs based on the number of weight levels is illustrated in Figure 3. This shows that the number of converged runs increases considerably with the number of weight levels. This trend is clearly confirmed by the results for the Sonar benchmarks (Table 10). In this table, the cases in which all runs converged, like for continuous pre-training (Table 5), have been set in bold font. For the Sonar benchmark at least four different weight levels (2 bits) are needed for results comparable to the continuous case, the LCLV4a curve being an exception for which at least 16 weight levels are needed. These results also illustrate that, in general, the number of epochs needed for training discrete networks decreases with the number of weight levels.

For the benchmarks that assess the generalization performance (Wine and Digit), the percentage of misclassification on the test set has been set in bold font, whenever it was at most two percent higher than for continuous pre-training. With such a small difference the performance is considered to be satisfactory and comparable to the one obtained when using continuous weights. The generalization results for the Wine benchmark are comparable with the results for the continuous network when the number of weight levels is at least six, with the LCLV3 curve as exception for which 16 levels are needed (which might again be caused by its high y-intercept). However, these results are biased by the occurrence of several runs, where the network is not able to learn at all. This problem caused by the discrete network getting trapped on an error plateau, is inherent to most weight discretization methods and might be resolved by introducing a random (annealing) factor in the weight updates [28]. To get a less biased idea of the possible performance, the average over the best five runs out of ten is listed in Table 11. In this case, the results are close to the continuous ones with the number of weight levels equal to 4 or 6. For LCLV2, the solutions with only four weight levels even show a lower misclassification rate than in the continuous case. This might be interpreted as an illustration of *Occam's razor* which states that simple models (for instance with discrete weights) should be preferred over more complicated ones.

The results for the Digit benchmark illustrate the challenging nature of this data set (Table 12). For most LCLV curves, at least 8 or 16 weight levels are needed to obtain classification results that are comparable to the continuous valued case. The on-line training with the LCLV3 curve is an exception

LCLV	Method	Weight Discretization Levels									
		2 levels		4 levels		6 levels		8 levels		16 levels	
		%Conv	#epochs	%Conv	#epochs	%Conv	#epochs	%Conv	#epochs	%Conv	#epochs
LCLV1	On-line	0.0	-	80.0	477.2	**100.0**	328.6	**100.0**	88.4	**100.0**	58.6
	Batch	0.0	-	60.0	2340.0	**100.0**	463.4	**100.0**	564.6	**100.0**	119.2
LCLV2	On-line	0.0	-	80.0	289.8	80.0	490.5	**100.0**	302.8	**100.0**	128.8
	Batch	0.0	-	40.0	3662.0	80.0	2166.0	80.0	1378.2	80.0	134.2
LCLV3	On-line	80.0	1095.0	**100.0**	130.4	**100.0**	23.4	**100.0**	17.0	**100.0**	8.8
	Batch	60.0	2241.7	**100.0**	794.2	**100.0**	151.2	**100.0**	51.4	**100.0**	13.6
LCLV4a	On-line	0.0	-	0.0	-	0.0	-	40.0	486.5	**100.0**	87.0
	Batch	0.0	-	0.0	-	0.0	-	20.0	1949.0	40.0	3065.5
LCLV4b	On-line	20.0	965.0	60.0	374.3	60.0	517.3	20.0	104.0	60.0	182.0
	Batch	0.0	-	60.0	969.0	60.0	579.7	80.0	239.0	**100.0**	135.6

Table 10: Discretization results from the simulations (average over 5 runs) with Sonar(8).

LCLV	Method	Percentage of Test Misclassification					
		Weight Discretization Levels					
		C	2	4	6	8	16
LCLV1	On-Line BP	3.18	7.27	**5.00**	**4.55**	5.45	5.45
	Batch BP	2.27	22.73	**3.18**	**3.18**	**2.73**	**3.64**
LCLV2	On-Line BP	4.55	**4.55**	**3.63**	**4.09**	**4.09**	**4.55**
	Batch BP	2.27	15.00	**1.82**	**3.64**	**3.18**	**4.09**
LCLV3	On-Line BP	2.27	59.09	9.09	**2.73**	**2.73**	**2.27**
	Batch BP	2.27	59.09	34.55	**2.73**	**2.73**	**3.18**
LCLV4a	On-Line BP	3.18	20.45	8.18	**2.73**	**2.27**	**2.27**
	Batch BP	3.18	11.82	**4.09**	**4.09**	**4.09**	**4.09**
LCLV4b	On-Line BP	2.27	4.09	**2.27**	**2.27**	**2.27**	**2.27**
	Batch BP	2.27	55.45	5.91	4.55	**3.18**	**2.73**

Table 11: Discretization results from the simulations (average over the best 5 runs out of 10) with Wine(6).

(due to its asymmetry) and even 64 weight levels are not sufficient in this case. It is interesting to note that for LCLV4a and LCLV4b, the discretization results with 64 weight levels the misclassification rate is even lower than for the continuous network, which offers yet another illustration of Occam's razor.

3.2.4 Experimental Results on an Optical Thresholding Perceptron

Initial experiments have been performed for (2-layer) perceptron recall using an optical system with LCTVs to implement the input layer and the interconnection weight matrices and LCLVs to implement non-linear thresholding (the architecture is described in more detail in Refs. 5&29). Training of the perceptron was done with our adapted training algorithm (including weight discretization, compensation for the LCLV non-linearities, and transformation of the weights to the all-positive domain) and was simulated on a computer. The data set used was a small subset of 50 digits from the Digit benchmark (inputs having 16 gray levels) and weights were discretized to 8 gray levels. The resultant discrete weights were implemented on the LCTV of the optical perceptron for testing recall with 10 of the 50 digits in the training set. The optical recall was satisfactory with quite a good agreement with the values obtained in the computer simulations [29].

A binary perceptron, trained using our adapted training rule, implemented with binary inputs and weights in the same optical system also showed satisfactory recall results [30]. The advantage of having binary weights and input values is that they permit the use of ferroelectric liquid crystals having thousandfold faster response times than nematic LCLVs. Binary neural networks also have the inherent advantages of digital vs. analog implementations.

LCLV	Method	Percentage of Test Misclassification						
		Weight Discretization Levels						
		C	2	4	8	16	32	64
LCLV1	On-line BP	6.32	21.92	12.32	11.76	9.36	**7.92**	**7.36**
	Batch BP	5.60	12.48	**6.96**	**6.24**	**6.48**	**6.08**	**5.68**
LCLV2	On-line BP	6.56	18.16	10.56	10.00	**7.92**	**6.72**	**6.64**
	Batch BP	6.24	11.84	**7.28**	**6.48**	**6.64**	**6.40**	**6.40**
LCLV3	On-line BP	10.24	52.88	27.44	23.68	29.76	20.08	14.24
	Batch BP	8.48	21.44	14.00	**10.08**	**8.88**	**8.96**	**8.72**
LCLV4a	On-line BP	6.16	12.56	9.52	**8.00**	**7.04**	**6.88**	**5.92**
	Batch BP	6.40	72.40	9.60	**7.12**	**7.04**	**6.48**	**6.32**
LCLV4b	On-line BP	6.24	47.60	9.52	**6.80**	**6.48**	**6.40**	**5.84**
	Batch BP	6.72	90.00	9.28	**6.96**	**7.12**	**6.32**	**6.40**

Table 12: Discretization results from the simulations (average over 5 runs) with Digit(64).

4 Conclusions

A flexible weight discretization algorithm has been combined with a transformation of the weights that enables all-positive forward propagation in optical multilayer neural networks. This algorithm allows successful training when discrete weights are used in hardware, such as implemented by liquid crystal television screens in optical neural networks. Moreover, we include our method to compensate for the differences between non-standard activation functions as available in hardware and the standard sigmoidal activation function. The feasibility of a discrete non-negative multilayer neural network using real activation functions as available in LCLV hardware is confirmed by the results of a series of computer simulations. These simulations have been performed on four benchmark problems and using the experimentally observed characteristics of five optical activation functions realized by off-the-shelf liquid crystal light valves. We show that a weight accuracy of 1 to 4 bits is sufficient to get highly satisfactory performance that is comparable with the results obtained when using continuous weights. The results of the computer simulations are confirmed by initial experiments on an optical system for perceptron recall on a small set of handwritten digits, that show good performance in agreement with the simulated values.

Acknowledgements

The authors would like to thank the Swiss National Science Foundation (FN:20-43220.95 and FN:21-45621.95) for their support of this research. This research was performed when all authors were with the Dalle Molle Institute of Perceptual Artificial Intelligence (IDIAP), Martigny, Switzerland.

Appendix: Weight Discretization Algorithm

A multilayer neural network can have an arbitrary number of layers. Adjacent layers are fully inter-layer connected. Here, the weight from neuron i to neuron j is denoted by w_{ij}, the bias of neuron j is denoted by θ_j, its activation value is indicated as a_j, and t_j denotes the target pattern value for output neuron j. The on-line backpropagation algorithm is described by the following five steps:

1. **Initialization** Weights and biases are initialized with random values [13].

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

3. **Forward propagation** During this phase, the activation values of the input neurons are propagated through the network. The input of a neuron j, not in the input layer, is defined as:

$$y_j = (\sum_i w_{ij} a_i) - \theta_j .$$

The new activation value of neuron j is:

$$a_j = \phi(y_j),$$

where ϕ is a differentiable activation function, for example, a sigmoid.

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:

$$\delta_j = (t_j - a_j)\phi_j'(y_j) \qquad \text{if } j \text{ is an output neuron,}$$

and

$$\delta_j = (\sum_k \delta_k w_{jk})\phi_j'(y_j) \qquad \text{if } j \text{ is not an output neuron.}$$

After the calculation of all error signals, the weights are updated with learning rate parameter η:

$$w_{ij} := w_{ij} + \eta \delta_j a_i .$$

5. **Convergence test** If no convergence, the next pattern is presented (goto **Pattern presentation**).

The weight discretization method starts by fully training the network with the backpropagation algorithm. Next, the continuous weights are discretized by mapping them to the closest discretization level using a staircase shaped multiple thresholding function. The so-created discrete weights are then used for the forward propagation pass through the network. The errors obtained, which are based on the difference between the actual and desired network outputs, are subsequently used to update the continuous valued weights during the backward propagation pass. Note that pre-training with continuous weights is not necessary, but is often included for faster convergence. The discretization of the weights described by Fiesler *et al.* [16] uses d discretization levels with $d = 2$, 3, 5, 7, or 9. The discretization levels are symmetric around zero, except for $d = 2$ and are equidistant:

$$\left\{ n - \left\lfloor \frac{d+1}{2} \right\rfloor \;\middle|\; n = 1, 2, ..., d \right\}.$$

References

[1] I. Saxena and P. Horan, "Optical Implementations," in *the Handbook of Neural Computation*, E. Fiesler and R. Beale, eds., chapter E1.3, Institute of Physics Publishers & Oxford University Press, New York (1997).

[2] F. Yu, T. Lu, X. Yang, and D. Gregory, "Optical Neural Network with Pocket-Sized Liquid-Crystal Televisions," *Optics Letters*, 15(15), 863–865, (1990).

[3] J.-S. Jang, S.-G. Shin, S.-W Yuk, S.-Y Shin, and S.-Y. Lee, "Dynamic Optical Interconnections Using Holographic Lenslet Arrays for Adaptive Neural Networks," *Optical Engineering*, 32(1), 80–87 (1993).

[4] R. G. Stearns, "Optically Programmed Neural Network Capable of Stand-Alone Operation," *Applied Optics*, 32(26), 5141–5152 (1993).

[5] I. Saxena and E. Fiesler, "Adaptive Multilayer Optical Neural Network with Optical Thresholding," *Optical Engineering,* special on Optics in Switzerland, P. Rastogi, ed., 34(8), 2435–2440 (1995).

[6] N. Collings, "Design Considerations for a Useful Two-Layer Neural Network," presented at the *Euro-American Workshop on Optical Pattern Recognition*, La Rochelle, France (June 1994).

[7] N. Collings, A. R. Pourzand, and R. Volkel, "Construction of a Programmable Multilayer Analogue Neural Network Using Space Invariant Interconnects," vol. SPIE 2565, pp. 40–47 (1995).

[8] P. Moerland, E. Fiesler, and I. Saxena, "Incorporation of Liquid-Crystal Light Valve Non-Linearities in Optical Multilayer Neural Networks," *Applied Optics*, 35(26), 5301–5307 (1996).

[9] D. Psaltis and Y. Qiao, "Adaptive Multilayer Optical Networks," in *Progress in Optics*, E. Wolf, ed., vol. 31, chap. 4, pp. 227–261, Elsevier Science Publishers, Amsterdam, The Netherlands (1993).

[10] W. Xue, "Characterization of Liquid Crystal Light Valves for Neural Network Applications," Ph.D. thesis, Institute of Micro-technology, University of Neuchâtel, Neuchâtel, Switzerland (1994).

[11] N. Collings and W. Xue, "Liquid Crystal Light Valves as Thresholding Elements in Neural Networks: Basic Device Requirements," *Applied Optics,* 33(14), 2829–2833 (1994).

[12] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations, chap. 8, pp. 318–362, MIT Press, Cambridge, Massachusetts (1986).

[13] G. Thimm and E. Fiesler, "Weight Initialization in Higher Order and Multi-Layer Perceptrons," *IEEE Transactions on Neural Networks*, 8(2), 349–359 (1997).

[14] I. Shariv, O. Gila, and A.A. Friesem, "All-Optical Bipolar Neural Network with Polarization-Modulating Neurons," *Optics Letters*, 16(21), 1692–1694 (1991).

[15] G. Thimm, P. Moerland, and E. Fiesler, "The Interchangeability of Learning Rate and Gain in Back-propagation Neural Networks," *Neural Computation*, 8(2), 451–460 (1996).

[16] E. Fiesler, A. Choudry, and H. J. Caulfield, "A Weight Discretization paradigm for Optical Neural Networks," in *Proceedings of the International Congress on Optical Science and Engineering*, vol. SPIE 1281, pp. 164–173, SPIE, Bellingham, Washington (1990).

[17] N. Kasama, Y. Hayasaki, T. Yatagai, M. Mori, and S. Ishihara, "Experimental Demonstration of Optical Three Layer Neural Network," *Japanese Journal of Applied Physics*, 29(8), L1565–L1568 (1990).

[18] M. Kranzdorf, B. Bibner, L. Zhang, and K. Johnson, "Optical Connectionist Machine with Polarization-Based Bipolar Weight Values," *Optical Engineering*, 28, 844–848 (1989).

[19] I. Saxena, E. Fiesler, and P. Moerland, "A Method for All-Positive Optical Multilayer Perceptrons," in *Proceedings of the Third IEEE International Conference on Electronics, Circuits, and Systems (ICECS'96),* vol. 1, pp. 448–451, IEEE, Piscataway, NJ (1996).

[20] 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*, 2(1), 110–117 (1991).

[21] M. G. Robinson and K. M. Johnson, "Noise Analysis of Polarization-Based Optoelectronic Connectionist Machines," *Applied Optics,* 31(2), 263–272 (1992).

[22] L. Prechelt, "PROBEN1 — A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms," Technical report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany (1994). The benchmark set is available at ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz.

[23] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, 1, 75–89 (1988).

[24] Data made available in 1994 by librarians P. M. Murphy and D. W. Aha from the UCI Repository of Machine Learning Databases, a machine-readable data repository accessible via anonymous-ftp: ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

[25] M. D. Garris and R. A. Wilkinson, NIST Special Database 3, National Institute of Standards and Technology, Advanced System Division, Image Recognition Group (1992).

[26] 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, USA (1988).

[27] M. Moreira and E. Fiesler, Neural Networks with Adaptive Learning Rate and Momentum Terms. IDIAP-RR 95-04, IDIAP, Martigny, Switzerland. ftp://ftp.idiap.ch/pub/reports/1995/95-04.ps.Z.

[28] T. Lundin and P. Moerland, Quantization and Pruning of Multilayer Perceptrons: Towards Compact Neural Networks. IDIAP-Com 97-02, IDIAP, Martigny, Switzerland. Available via anonymous ftp: ftp://ftp.idiap.ch/pub/reports/1997/com97-02.ps.gz.

[29] I. Saxena, P. Moerland, E. Fiesler, A. R. Pourzand, and N. Collings, "An Optical Thresholding Perceptron," in *Proceedings of the Workshop on Optics and Computer Science*, Geneva, Switzerland (1997).

[30] I. Saxena, P. Moerland, E. Fiesler, and A. Pourzand, "Handwritten Digit Recognition with Binary Optical Perceptron," in *Artificial Neural Networks - ICANN'97*, Lecture Notes in Computer Science, vol. 1327, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, eds., pp. 1253–1258, Springer Verlag, Berlin (1997).