Published in the Proceedings of the
'96 SIPAR-Workshop on Parallel and Distributed Computing
Genève, Switzerland, October 4, 1996.

Connectionist Quantization Functions

T. Lundin, E. Fiesler and P. Moerland

IDIAP, CP 592, CH-1920 Martigny, Switzerland, E-mail: Tomas.Lundin@idiap.ch

One of the main strengths of connectionist systems, also known as neural networks, is their massive parallelism. However, most neural networks are simulated on serial computers where the advantage of massive parallelism is lost. For large and real-world applications, parallel hardware implementations are therefore essential. Since a discretization or quantization of the neural network parameters is of great benefit for both analog and digital hardware implementations, they are the focus of study in this paper. In 1987 a successful weight discretization method was developed, which is flexible and produces networks with few discretization levels and without significant loss of performance. However, recent studies have shown that the chosen quantization function is not optimal. In this paper, new quantization functions are introduced and evaluated for improving the performance of this flexible weight discretization method.

Keywords: *(artificial) neural networks, connectionist systems, weight discretization, quantization, generalization, hardware implementation.*

1 Introduction

A connectionist system or neural network is a massively parallel network of weighted interconnections which connect one or more layers of non-linear processing elements (*neurons*). To fully profit from the inherent parallel processing of these networks, the development of parallel hardware implementations is essential. However, these hardware implementations often differ in various ways from the ideal mathematical description of a neural network model. It is, for example, required to have quantized network parameters, in both electronic and optical implementations of neural networks. This can be because device operation is quantized or a coarse quantization of network parameters is beneficial for reducing VLSI surface area.

Most of the standard algorithms for training neural networks are not suitable for quantized networks because they are based on gradient descent and require a high accuracy of the network parameters: 8 bits are required for on-chip recall and 16 bits for on-chip training [Holt-93]. Several weight discretization techniques have been developed to reduce the required accuracy further without deterioration of network performance. One of the earliest and perhaps most successful of these techniques [Fiesler-88] is further investigated and improved in this paper. The basic idea of this method is to first train the neural network with the backpropagation algorithm without quantization of the weights. Secondly, the resulting continuous weights are discretized by mapping them to the closest discretization level. These discrete weights are then used in the forward propagation pass through the network. The resulting errors, which are based on the difference between the obtained and desired network outputs, are subsequently used to update the continuous weights during the backward pass until satisfactory performance is obtained. In the original paper [Fiesler-88] a staircase shaped threshold function with a uniform distribution is used to perform the mapping to equidistant discrete weights.

In this paper, five alternative quantization functions are introduced and compared with the original one in a series of experiments on five real-world benchmark problems. The main goal is to improve the original quantization function as much as possible to obtain networks with both a small number of discrete weight levels and good generalization performance on unseen test data. Therefore, our attention is not restricted to quantization functions that are well suited for hardware implementation. However, the experimental results indicate that the three "hardware-friendly" quantization functions show the best results.

2 Weight Discretization Algorithm

In this section, a brief description of the six weight quantization functions, their inclusion in the weight discretization algorithm, and a discussion of their suitability for hardware implementation are presented. The original weight discretization algorithm maps the continuous weights to discrete weight levels, obtained from a quantization function, before each forward pass through the network. The weights are discretized by mapping them to the discrete weight levels to which they are closest. In order for the quantization functions to be flexible, a parameter D is included that indicates the number of weight discretization levels. The number of levels used in the experiments described in this paper are subsequently 2, 3, 5, 7, 15, and 31. These values have been chosen based on the fact that in digital hardware implementations generally $2^D - 1$ discretization levels are available ($D-1$ bits plus a sign bit). The cases of $D=2$ and $D=5$ have been added to obtain a more precise idea of the effects of having only a small (or even minimal for $D=2$) number of discretization levels.

2.1 Description of Quantization Functions

Six weight quantization functions have been implemented and tested. First some general notations are defined which are useful in the definition of these functions: W_+ is the maximum positive weight value and W_- is the minimum negative weight value of the pre-trained continuous network. The maximum absolute value of W_- and W_+ is denoted by W_{max}. The expected value (mean) of the weights calculated over the entire pre-trained continuous network is indicated as $E(w)$. The following quantization functions have been implemented and tested:

Symmetrical This is the original quantization function from [Fiesler-88]. The resulting discretization levels are symmetric around zero and equidistant, with a step size of one between the weight levels.

The intuitive disadvantage of the symmetrical quantization function is that it does not incorporate any knowledge about the weights of the pre-trained continuous network. This information is used in the new quantization functions such that the mapping to discrete weights does not completely perturb the results of continuous pre-training:

W_{max} This quantization function divides the interval $[-W_{max}, +W_{max}]$ in equidistant levels, resulting in weight levels which are symmetric around zero.

W_{max_adapt} This is a straightforward modification of the previous function. It consists of two different phases and uses both W_- and W_+. In the first phase the interval $[W_-, 0]$ and $[0, W_+]$ are divided in equidistant levels. The second phase consists of attributing the value zero to the level that is closest to zero.

In [Bellido-93] it is concluded that the weight distribution in a neural network resembles a Gaussian-like distribution. This means that there are many weights with small values and only few weights with large values. The power-of-two series, which is an inaccurate but simple approximation for a Gaussian distribution, is used in this paper to approximate the weight distribution of the pre-trained network:

Power_of_two_W_{max} This quantization function uses the interval $[-W_{max}, 0]$ and $[0, +W_{max}]$ which is divided using powers-of-two, that is, the weight levels are $0, \pm W_{max}/2^i$ for $i=0,1,...,\lfloor (D-1)/2 \rfloor$.

Power_of_two Instead of only using the W_{max} value this function divides both intervals, $[-W_{max}, E(w)]$ and $[E(w), W_{max}]$, using powers-of-two.

Power_of_two_adapt This is a modification of the previous quantization function which divides both intervals $[W_-, E(w)]$ and $[E(w), W_+]$ using powers-of-two.

Not all of the described quantization functions are equally well suited for hardware implementation. Only the quantization functions which result in weight levels that are symmetric around zero and equidistant, are suitable. These conditions are satisfied by the *Symmetrical*, W_{max}, and *Power_of_two_*W_{max} quantization functions. In this case the discrete weights can namely be normalized to the interval [-1,1] by dividing them by the maximal weight value. The normalized weights can then be encoded as binary numbers. The scaling of the weights can be compensated for by rescaling the gain (steepness) of the activation function.

benchmark	network topology[†]	pattern set sizes			input range	output range
		train.	val.	test		
Auto-mpg	7-3-1	196	98	98	[0,1]	[0,1]
Sunspot	12-2-1	105	52	52	[0,1]	[0,1]
Wine	13-6-3	89	44	45	[0,1]	[-1,1]
Cancer	9-6-2	350	174	175	[0,1]	[-1,1]
Diabetes	8-6-2	384	192	192	[0,1]	[-1,1]

Table 1: Summary of the benchmarks problems characteristics.

The *Power_of_two_*W_{max} quantization function has the added advantage that the normalized discrete weight values are restricted to powers-of-two. Therefore, simple shift registers can be employed to substitute the more complex multipliers [Marchesi-93].

The other three quantization functions W_{max_adapt}, *Power_of_two*, and *Power_of_two_adapt* lead to weight levels that are not symmetric around zero and are therefore not suited for hardware implementation.

3 Experiments and Results

To evaluate the performance of the six quantization functions, a series of experiments on five real-world benchmark problems has been performed. Each of the experiments consists of 10 different runs of the weight discretization algorithm using one of the quantization functions on each benchmark. The characteristics of these benchmarks and some of the network parameters are listed in Table 1. These benchmarks have been used to assess the generalization performance on a test set which is not used during training. A cross-validation technique with a training, validation, and test pattern set was used to decide when to stop training.

In all the experiments a multilayer perceptron, with only one hidden layer which is fully interlayer connected to both input and output layer, has been used. The non-linear activation function employed is a hyperbolic tangent, while for the Auto-mpg and Sunspot benchmark a linear activation function has been used in the output layer of the network. The desired output values for these two benchmarks are namely real-valued (approximation problem), while the other benchmarks are classification problems with desired output values of -1 and $+1$. For each run the network was initialized with different random weights in the interval [-0.77,0.77], which has been chosen according to [Thimm-96]. Furthermore, a learning rate of 0.5, a momentum term of 0.9, and a flat-spot constant of 0.1 have been used.

Additional information about the different benchmarks and the experimental set-up can be found in [Lundin-96].

3.1 Discussion of Results

To evaluate the experimental results, two different performance measurements have been used. For the approximation problems (Auto-mpg and Sunspot), the normalized mean square error on the test set is most significant. For the classification problems (Wine, Cancer and Diabetes), the percentage of misclassified test patterns has been used. A pattern is considered correctly classified whenever the highest network output corresponds to the correct class.

First, it was observed that the results depend on the type of benchmark problem used. In fact, the results can be divided into two classes: one for the classification problems and the other for the approximation problems. Therefore, the results of these two classes are discussed separately. Due to lack of space, the results for only two benchmarks problems are listed in tabular form: the Diabetes benchmark (Table 2) being representative for the classification problems and the Auto-mpg benchmark (Table 3) for the approximation problems.

A ranking system was used to evaluate the outcome of the experiments. For each benchmark and discretization level, the test set results of the different quantization functions were ranked from one to

[†]# of neurons in the input-hidden-output layer.

Discretization function	D	# of epochs (mean)	% of misclassification (mean)		
			train.	val.	test
	C[‡]	107.5	19.97	25.89	23.49
Symmetrical	2	216.0	24.66	30.78	25.26
	3	172.5	24.51	30.16	25.10
	5	57.5	25.91	29.79	27.14
	7	49.0	25.89	29.22	27.45
	15	54.5	23.28	28.39	26.25
	31	82.0	23.20	27.66	25.99
W_{max}	2	56.5	28.07	29.43	30.16
	3	57.5	26.43	32.08	29.64
	5	64.5	22.94	28.02	25.26
	7	67.5	22.81	27.29	27.03
	15	98.0	20.00	26.30	24.84
	31	136.5	20.76	26.41	24.90
W_{max_adapt}	2	5.0	33.59	36.46	35.94
	3	37.5	31.69	35.62	34.74
	5	93.5	25.62	29.69	29.06
	7	94.5	22.99	27.92	25.83
	15	98.0	20.62	25.89	25.26
	31	174.0	21.17	26.82	25.05
Power_of_two _W_{max}	2	56.5	28.07	29.43	30.16
	3	57.5	26.43	32.08	29.64
	5	64.5	22.94	28.02	25.26
	7	54.5	21.90	27.76	26.35
	15	137.0	20.52	25.68	24.22
	31	82.0	21.33	26.56	26.04
Power_of_two	2	56.5	28.07	29.43	30.16
	3	59.0	33.26	35.31	34.53
	5	62.0	24.48	27.81	26.98
	7	79.5	20.86	26.15	24.01
	15	73.5	21.20	27.24	25.21
	31	75.0	21.82	26.09	24.74
Power_of_two _adapt	2	86.5	27.01	29.01	28.80
	3	80.0	30.49	33.07	32.60
	5	61.0	24.01	27.97	27.45
	7	81.5	20.96	26.30	24.48
	15	100.0	21.22	26.46	25.68
	31	104.5	21.51	26.77	25.52

Table 2: Discretization results for the *Diabetes* benchmark.

Discretization function	D	# of epochs (mean)	square error % (mean)		
			train.	val.	test
	C	171.5	0.36	0.31	0.24
Symmetrical	2	5.0	8.08	8.61	7.04
	3	5.0	8.08	8.61	7.04
	5	8.5	6.55	6.98	5.66
	7	10.0	5.30	5.65	4.55
	15	12.0	2.06	2.07	1.71
	31	9.0	0.46	0.36	0.39
W_{max}	2	9.0	1.69	1.49	1.64
	3	35.5	1.18	1.10	1.21
	5	48.5	0.58	0.45	0.45
	7	8.0	0.43	0.39	0.33
	15	77.5	0.44	0.34	0.32
	31	108.5	0.38	0.31	0.27
W_{max_adapt}	2	9.0	2.22	2.26	2.27
	3	29.5	0.99	0.90	0.96
	5	61.0	0.63	0.48	0.54
	7	53.0	0.49	0.42	0.39
	15	63.5	0.42	0.32	0.31
	31	137.0	0.38	0.30	0.27
Power_of_two _W_{max}	2	9.0	1.69	1.49	1.64
	3	35.5	1.18	1.10	1.21
	5	48.5	0.58	0.45	0.45
	7	36.0	0.50	0.38	0.40
	15	33.5	0.40	0.33	0.27
	31	37.0	0.39	0.33	0.27
Power_of_two	2	9.0	1.69	1.49	1.64
	3	19.5	1.14	1.03	1.07
	5	30.5	0.72	0.57	0.63
	7	45.0	0.51	0.38	0.39
	15	27.0	0.44	0.35	0.32
	31	32.5	0.46	0.35	0.35
Power_of_two _adapt	2	21.0	1.62	1.46	1.52
	3	44.0	1.01	0.85	0.94
	5	35.5	0.62	0.47	0.49
	7	40.0	0.51	0.40	0.39
	15	29.0	0.43	0.36	0.31
	31	23.5	0.43	0.36	0.33

Table 3: Discretization results for the *Auto-mpg* benchmark.

six. The ranking scores were then added within each quantization function and class (approximation or classification). This gives an overall performance for all quantization functions in each class.

As is illustrated in table 2, the *Symmetrical* quantization function performs good for classification problems using only two or three discretization levels. In fact, for the Cancer and Diabetes benchmarks the same performance is not obtained by any of the other quantization functions until using seven or more discretization levels. For example, using $D = 3$, the percentage of misclassification for the *Symmetrical* quantization function is 25.10 (rightmost column of Table 2). The *Power_of_two* quantization function is the first to produce a better result with $D = 7$. Of the five new quantization functions the *Power_of_two*, *Power_of_two_*W_{max}, and W_{max} quantization functions perform best. For these functions, $D = 7$ is sufficient for both power-of-two based quantization functions, while the results for W_{max}, using $D = 15$, are comparable with those from the continuous pre-training. Less good performance was noticed for the W_{max_adapt} and *Power_of_two_adapt* quantization functions.

For the approximation problems, the *Symmetrical* and to a less extent the *Power_of_two* quantization functions give an overall poor performance. Note, that this differs from the results for the classification problems. The *Power_of_two_*W_{max},W_{max} and W_{max_adapt} quantization function perform best and results for $D = 15$ are comparable with the ones from continuous pre-training. For example, using 15 discretization levels the *Power_of_two_*W_{max} quantization function gives a mean square error percentage of 0.27 (rightmost column of Table 3), which is almost as good as in the continuous case. The *Power_of_two_adapt* performs

[‡]'C' denotes the results for the continuous pre-training.

well on the Auto-mpg problem, but less good on the Sunspot problem.

4 Conclusions

The performance of six different quantization functions, in training multilayer perceptrons with a small number of discrete weight levels, has been evaluated.

Two of these quantization functions, W_{max} and $Power\_of\_two\_W_{max}$, show generally good results in a series of experiments including both classification and approximation problems. For these quantization functions, only 15 discrete weight levels are sufficient to obtain nearly the same performance as for a network with continuous weight values. Both quantization functions are also well suited for hardware implementation, for example by using only shift registers when $Power\_of\_two\_W_{max}$ is implemented.

The results also indicate that for classification problems the *Symmetrical* quantization function performs good when using only ternary weights $\{-1, 0, 1\}$ or even binary weights $\{-1, 1\}$.

Acknowledgements

The authors would like to thank the Swiss National Science Foundation for their partial support of this research (FN:2100-045621.95/1).

References

[Bellido-93] I. Bellido and E. Fiesler. Do Backpropagation Trained Neural Networks have Normal Weight Distributions? *Proc. of the Int. Conf. on Artificial Neural Networks*, pp. 772–775, Springer, London, 1993.

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

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

[Lundin-96] T. Lundin, E. Fiesler, and P. Moerland. Quantization Functions for Multilayer Perceptrons. Technical report 96-xx (in preparation), IDIAP, Martigny, Switzerland 1996.

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

[Thimm-96] G. Thimm and E. Fiesler. Weight Initialization in Higher Order and Multi-Layer Perceptrons. Accepted for publication in *IEEE Transactions on Neural Networks*, 1996.