Experimental Determination of Precision Requirements for Back-Propagation Training of Artificial Neural Networks

Krste Asanović
Nelson Morgan

TR-91-036

October 1991

Abstract

The impact of reduced weight and output precision on the back-propagation training algorithm [Wer74, RHW86] is experimentally determined for a feed-forward multilayer perceptron. In contrast with previous such studies, the network is large with over 20,000 weights, and is trained with a large, real-world data set of over 130,000 patterns to perform a difficult task, that of phoneme classification for a continuous speech recognition system.

The results indicate that 16b weight values are sufficient to achieve training and classification results comparable to 32b floating point, provided that weight and bias values are scaled separately, and that rounding rather than truncation is employed to reduce the precision of intermediary values. Output precision can be reduced to 8 bits without significant effects on performance.

1 Introduction

Research into artificial neural networks (ANNs) is hindered by their extensive computational demands. However, these algorithms exhibit massive fine-grained parallelism and require only moderate arithmetic precision. These properties can be exploited in the design of high performance, low cost neurocomputers.

Advances in CMOS VLSI technology are rapidly increasing the computational power that can be integrated on a single die. However, a VLSI neurocomputer with adequate processing power and storage capacity for current research problems will require hundreds, if not thousands, of dice with current technology. Inter-chip connections are much less dense than intra-chip connections, and inter-chip connections have much higher parasitic loads than intra-chip connections. These effects combine to ensure that for large multi-chip systems, sustainable performance is limited by inter-chip bandwidth, rather than by on-chip processing power.

Cost considerations dictate the use of commercial RAM packages to provide storage for all but the smallest systems, and so a significant fraction of the inter-chip bandwidth requirement will be for processor-memory traffic. The storage requirements of an ANN algorithm are dominated by storage for the connection weights, and the transfer of these weight values dominates processor-memory traffic. These observations imply that using the minimum weight precision necessary for satisfactory training and classification performance is essential in maximizing the performance of any large neurocomputing system.

During the execution of an ANN algorithm partitioned over a parallel neurocomputing system, the transfer of neuron output values dominates the traffic between processing nodes. Using the minimum necessary output precision is therefore important in reducing inter-processor communication requirements.

Reducing operand precisions has the added benefit of reducing the size of datapath circuitry. Though this effect is significant in reducing system costs with current technologies, the impact of future technology scaling will make the reduction of datapath widths much less important than the reduction in operand bandwidth across chip boundaries.

In this paper, the impact of reducing weight and output precisions is experimentally determined for a feed-forward multi-layer perceptron that forms part of a speech recognition system. The ANN is trained using the popular error back-propagation algorithm [Wer74, RHW86]. Earlier studies have examined artificial problems or small real data sets to determine acceptable ranges for these precisions. This paper contains an experimental analysis of a large network trained with a large real-world data set to perform a difficult task.

2 ANN Architecture

A phoneme-based speaker dependent continuous speech recognition system is under development at ICSI. The system utilizes a layered ANN to generate emission probabilities for a hidden Markov model (HMM) speech recognizer. Initial experiments indicate that this method compares favourably with conventional HMM speech recognition methods [MB90].

The network has 26 inputs. These are 12th order Perceptual Linear Prediction (PLP) coefficients [Her90] plus first order derivative terms for each speech frame. The input values are normalized to zero-mean and unity variance across the set of training data. These inputs are directly and fully connected to a layer of 256 hidden units. There are 61 outputs from the network, one per DARPA-phone to be recognized. The hidden layer is directly and fully connected to the output layer.

The database used consists of 500 sentences of continuous read speech from speaker DTD of the Resource Management RM-1 corpus, totalling 166023 10ms frames. Each frame is annotated with a phonetic label derived from an iterative application of the Viterbi algorithm on a path constrained by the correct transcription of the training set. The data is split into a training set of 400 sentences (131322 frames) and a test set of 100 sentences (34701 frames). The training set is used for the back-propagation learning algorithm. The test set is used to check the performance of the training algorithm to avoid over-fitting of the network to the training set.

There are two phases in the training scheme, which is a recent variant of the cross-validation learning approach used in [MB90]. Training starts by assigning some range of random values to all the weights, say $\pm r$. An initial value for α, the learning constant, is chosen and this value remains constant throughout the first training phase. A single training iteration consists of one pass through the training set, updating weights after every frame. After each training iteration, the net's performance is measured using the test set. When the performance improvement is less than some minimal improvement parameter $i\%$, the training scheme switches to the second phase. In the second phase of training, the value of α is halved before each iteration. When the performance improves by less than $i\%$ during the second phase, training stops.

The default values chosen for the training schedule parameters are initial random weights in the range $r = \pm 0.05$, initial $\alpha = 0.1$, and minimum performance improvement $i = 0.5\%$. Training performance has been found to be relatively insensitive to the exact value of these parameters.

3 Experimental Methods

All simulations were executed on various configurations of the RAP neurocomputer [MBAB90], the largest containing 24 processors. The RAP uses TMS320C30 DSPs that implement 32-bit floating-point arithmetic using a proprietary format [Tex88].

Reduced precision weight representations were simulated by adapting an existing ANN training program, "`mlp`", to call a weight quantization routine after each training pattern. The quantization routine rounds each updated floating-point weight to the nearest value allowed by the simulated fixed-point representation, saturating if necessary. The effect of this is to simulate a processor that stores weights to some lesser precision between training patterns, but which performs all other arithmetic using the DSP's native 32-bit floating-point precision. This provides an upper bound on the performance to be expected with lower precision weights, as in practice a neurocomputer may also implement other parts of the algorithm using lower precision arithmetic. The TMS320C30's floating point format has a 24-bit significand, represented as a normalized two's-complement number with an implied most-significant non-sign bit. This allows the simulations to model two's-complement weights of up to 25 bits, where this number includes the sign bit.

The original `mlp` program used a table lookup with over 1000 entries to implement the sigmoid function. Reduced output precisions were simulated by quantizing the entries in this table.

4 Results

The first series of experiments investigated the effect of changing the exponent of the fixed-point weights. The exponent of the MSB (sign bit) was varied in the range 0–5 for weight precisions of 12, 16, 20, and 25 bits. For example, an MSB exponent of 1 for a 16-bit weight would allow weights in the range ± 2. Biases were treated the same as weights. The default parameters listed above were used for the training schedule and outputs were kept at full precision. The network had 256 hidden units giving a total of 22589 weights and biases. The results are presented in Figure 1. The graph plots the highest score achieved on the test set against the exponent of the MSB for each of the weight

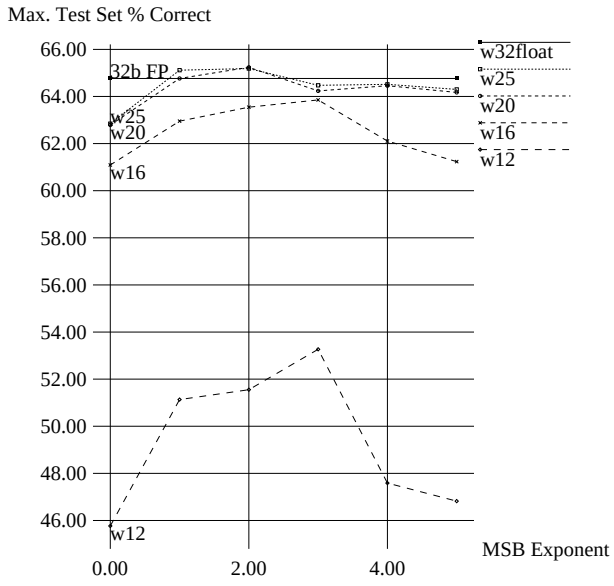

Figure 1: Effect of Varying Exponent and Precision. 256 hidden units, $\alpha = 0.01$, $r = \pm 0.05$, $i = 0.5\%$.

precisions [1].

The scores for 20b and 25b fixed point weights are comparable to those of the 32b floating point for MSB exponents in the range 1–5. The scores for the lower precision weights show much greater variation with MSB exponent. With 16b weights an MSB exponent of 3 provided the best scores, close to those of 32b floating point. The 12b weight runs also had the best performance with an MSB exponent of 3, though this gave over 10% lower scores than 32b floating-point.

The next series of experiments investigated a much larger number of weight precisions over a smaller range of MSB exponents. Weights in the range 12–20 bits and 25 bits were employed with MSB exponents in the range 1–3. The results are given in Figure 2, plotted as best test score against number of weight bits for different MSB exponents. Overall, the scores are fairly insensitive to changes in the weight MSB exponent in the range 1–3, although lower precision weights (12–16b) have slightly better scores with a range of ± 8, while higher precision weights (19–25b) have slightly better scores with a range of ± 4. These results would seem to indicate that around 17–18 bits are required to achieve scores comparable to that of 32b floating-

[1] The best scores in these experiments are somewhat lower than the scores we have reported recently [MHB+91], but these latter studies used networks with 200,000–300,000 weights, and still only achieved about 6% better performance at the frame level than the nets used in the current experiment. These latter nets yield a final word performance for the perplexity-60 word pair grammar of over 92%, a reasonable score for a simple context-independent HMM system.

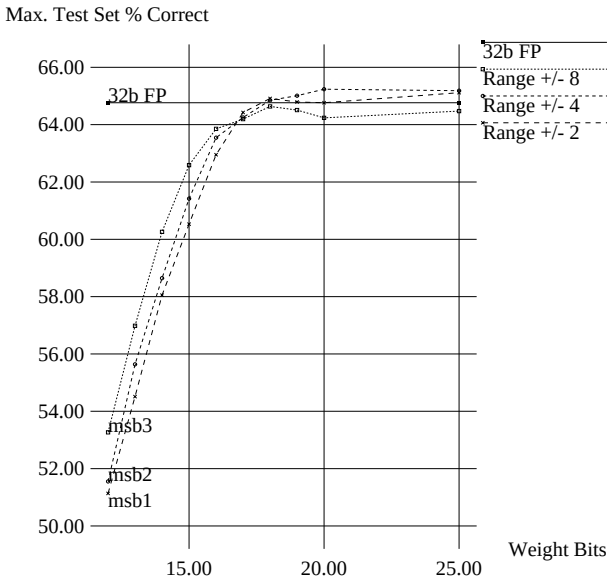

Figure 2: Effect of Varying Exponent and Precision. 256 hidden units, $\alpha = 0.01$, $r = \pm 0.05$, $i = 0.5\%$.

point, if the weights and biases are treated uniformly.

It was found that the higher scoring networks took roughly the same number of iterations (8–12) over the training set to converge as the 32b floating point version (9). Lower scoring networks tended to peter out in fewer training iterations (4–6). This indicates that there is no loss in training performance, as measured by number of training presentations required, for lower precision weights.

The weight and bias distributions resulting from these first experiments were examined. Figure 3 shows a bias histogram for the 32b floating point network after training, Figure 4 shows the weight histogram. Figures 5 and 6 show bias and weight distributions for a 16b network with weights in the range ± 8.

These histograms show that weight values tend to be clustered around zero in a Gaussian distribution, while bias values tend to be larger and negative. In the 16b net the bias values are seen to be hitting against the range limitation. These histograms suggested that it would be beneficial to fix the bias exponent separately from the weight exponent.

In the next set of experiments, the bias exponent was varied over the range 2–5 while the weight exponent was set to 1–2. Figure 7 plots the results for weights in the range ± 2, and Figure 8 plots the results for weights in the range ± 4. For short weight lengths, scaling the bias values separately gives a large improvement in scores. Higher weight precisions were fairly insensitive to changes in the bias exponent. These results show that 16b weights in the range ± 2

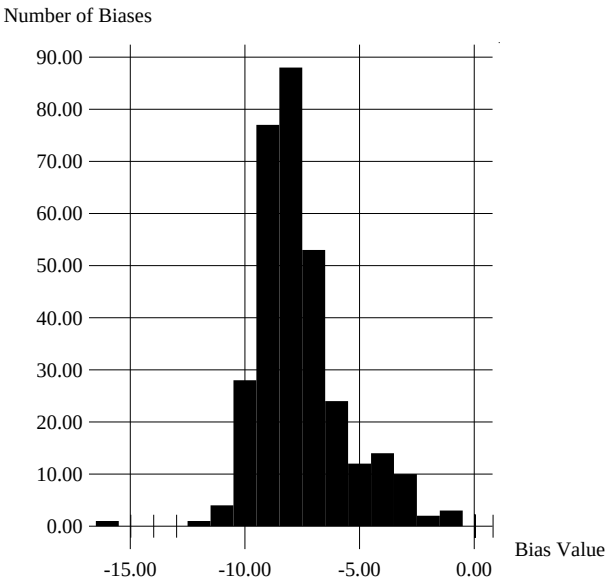

Figure 3: Histogram of 32b float biases of trained network.

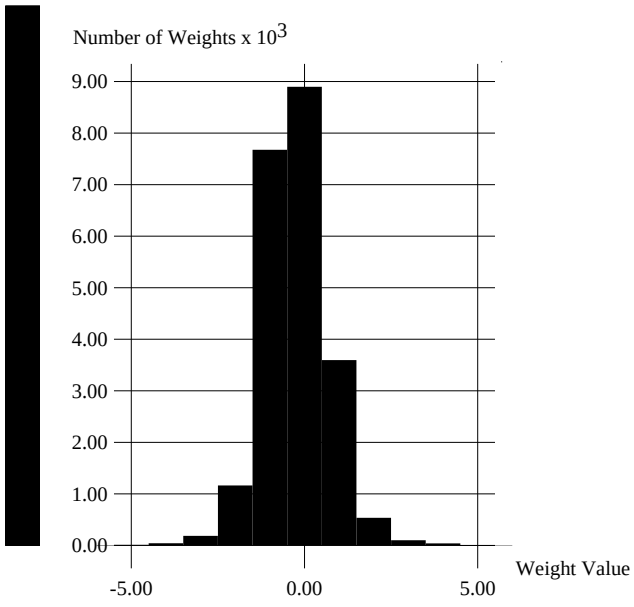

Figure 4: Histogram of 32b float weights of trained network.

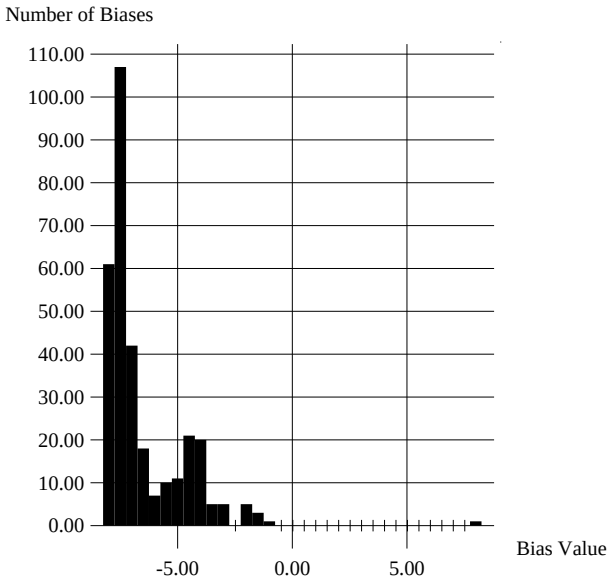

Figure 5: Histogram of 16b biases of trained network. Biases rounded to range ±8.

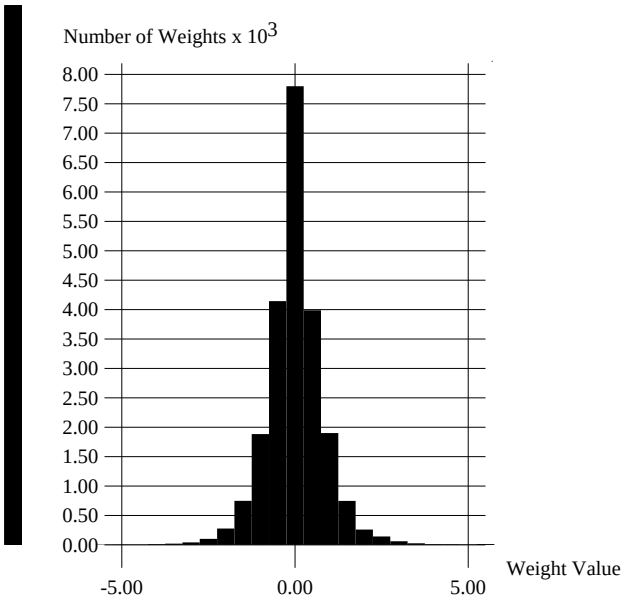

Figure 6: Histogram of 16b weights of trained network. Weights rounded to range ±8.

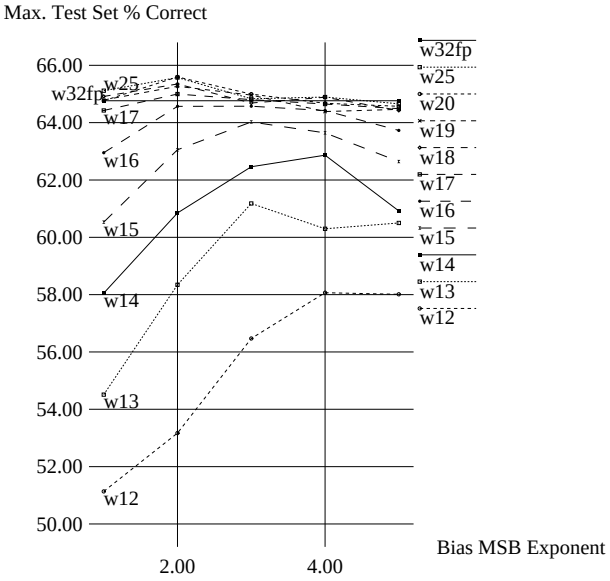

Figure 7: Altering Bias Scaling. Weights in range ±2.

with bias values in the range ±4–±16 give results comparable to 32b floating-point.

The results so far have used full precision for the output values. A series of simulations were performed reducing the precision of the output values. Figure 9 shows results for a network with 25b weights of range ±2 with biases of range ±4, and for a network with 16b weights of range ±2 with biases of range ±8. These scalings gave the best results for each weight precision in the previous experiments. Output precisions of 1, 2, 4, 8, and 16 and 25 bits were measured.

The results show that 16b weights and biases with 8b outputs give essentially the same scores as using 32b floating-point values throughout. Using less than 4 bits for the output precision gives noticeably poorer performance for both 16b and 25b networks.

Initial experiments revealed that truncating intermediary weight values to the reduced precision produced much poorer results than using rounding. In Figure 10 truncation and rounding results are plotted for various weight precisions. All weights and biases were rounded in the range ±2 and outputs were kept at full precision. Note that there are more lower precision data points for the rounding curve, and more higher precision data points for the truncation curve. It can be seen that use of truncation reduces performance by an amount corresponding to roughly 6 bits of precision when compared to rounding. This demonstrates the sensitivity of the back-propagation algorithm to the quality of the arithmetic employed. Examination of the trained weight histograms revealed that the truncated weight networks were in a very different region of weight space than the floating point

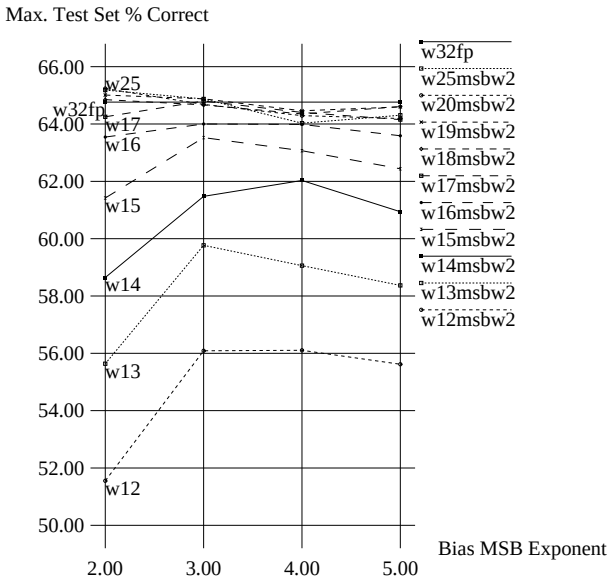

Figure 8: Altering Bias Scaling. Weights in range ± 4.

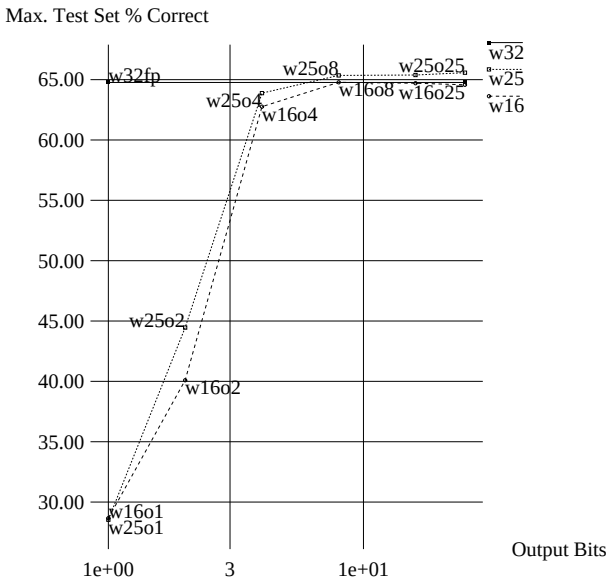

Figure 9: Varying Output Precision. 25b and 16b weight networks.

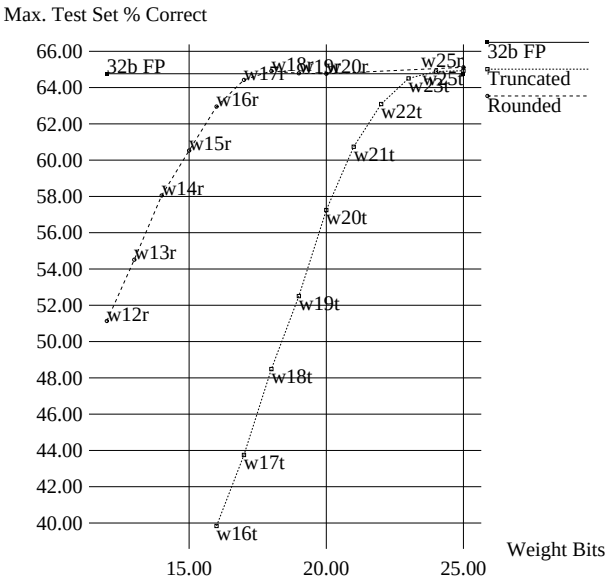

Figure 10: Truncation versus Rounding.

network. This can be explained by noting that truncation adds an overall negative bias to the gradient descent causing the network to follow a very different learning trajectory, reaching a poorer local minimum. All other results in this paper were obtained using rounding to reduce the precision of intermediary values.

5 Conclusions

We have shown that in the back-propagation training of a large, real-world neural net application it is possible to achieve essentially the same performance using 16b fixed-point weights and 8b fixed-point outputs instead of 32b floating-point values. This agrees with other researcher's findings with smaller networks and training sets [BH88]. Lower weight precisions gave significantly poorer results.

Achieving this performance required careful attention to the properties of short word length arithmetic. Truncation was shown to give very poor results; rounding should be used to reduce the precision of intermediary results. Scaling bias values separately from weight values provides a noticeable increase in performance for short word length weights.

6 Acknowledgements

Thanks to Phil Kohn who wrote the original `mlp` training program and who modified this to incorporate the fixed-point routines. The National Science Founda-

tion has provided explicit support for this project with Grant No. MIP-8922354. We also gratefully acknowledge the support of the International Computer Science Institute.

References

[BH88] T. Baker and D. Hammerstrom. Modifications to artificial neural network models for digital hardware implementation. Technical Report CS/E 88-035, Department of Computer Science and Engineering, Oregon Graduate Center, 1988.

[Her90] H. Hermansky. Perceptual Linear Predictive (PLP) Analysis of Speech. *J. Acoust. Soc. Am.*, 87(4), April 1990.

[MB90] N. Morgan and H. Bourlard. Continuous speech recognition using Multilayer Perceptrons with Hidden Markov models. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing*, pages 413–416, Albuquerque, New Mexico, USA, 1990.

[MBAB90] N. Morgan, J. Beck, E. Allman, and J. Beer. RAP: A Ring Array Processor for Multilayer Perceptron applications. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing*, pages 1005–1008, Albuquerque, New Mexico, USA, 1990.

[MHB+91] N. Morgan, H. Hermansky, H. Bourlard, P. Kohn, and C. Wooters. Continuous speech recognition using PLP analysis with Multilayer Perceptrons. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing*, pages 49–52, Toronto, Canada, 1991.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing. Exploration of the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[Tex88] Texas Instruments, Houston, Texas, USA. *Third-Generation TMS320 User's Guide*, 1988.

[Wer74] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Dept. of Applied Mathematics, Harvard University, 1974.