Proceedings of the IASTED
International Symposium

MACHINE LEARNING AND NEURAL NETWORKS

New York, U.S.A.
October 10 - 11, 1990

Editor : M. H. Hamza

A Publication of
The International Association of Science
and Technology for Development - IASTED

ISBN: 0-88986-172-2

ACTA PRESS
ANAHEIM * CALGARY * ZURICH

Application of Parallelized Back Propagation Algorithms to Speech Recognition

Brian K. Mak[1]
Speech Technology Laboratory,
3888 State Street,
Santa Barbara, CA 93105

Ömer Eğecioğlu
Department of Computer Science,
University of California,
Santa Barbara, CA 93106

Abstract

The performance of the parallel variants of the Back Propagation training algorithm for feed-forward neural networks is evaluated on two medium size speech problems: vowel classification, and English text-to-speech conversion (NETtalk data). Both the batch and non-batch modes of the training method are tested. The Back Propagation training algorithm itself is parallelized in three different ways, making use of different communication schemes of the hypercube multiprocessor. Evaluation is done on the following aspects: efficiency of parallelization, effect of processor size, the number of units of the neural net, and continuous learning. We show that the different communication methods affect the locality of data, the extent of relaxation, and the number of messages exchanged.

Keywords — Neural models, speech recognition, parallel training algorithm.

1 Introduction

Massive parallelism is believed to be essential for achieving human-like performance in fields such as speech and image recognition. Simulation of massive parallelism can be considered in at least three contexts: parallelism in modeling, computation, and algorithm design. Artificial Parallel Distributed Processing (PDP) neural network models attempt to encapsulate the model parallelism. On the other hand, computation parallelism can now be attained in many commercially available parallel machines.

In this paper we concentrate on utilizing different communication schemes of a hypercube multiprocessor (Intel iPSC/2), to parallelize the (non-batch) Back Propagation (BP) training algorithm and its corresponding (batch) Steepest Descent (SD) version for feed-forward neural networks. The performance of these parallel algorithms on two speech related problems, vowel classification and English text-to-speech conversion is evaluated. The design issues of the resulting alternate methods center around enhancing computational parallelism, providing relaxation, and minimizing communication costs on message-passing multiprocessors.

2 Parallelized and Modified Back Propagation Algorithms

The PDP model that we consider is a multi-layer perceptron-like feed-forward nonlinear network. The bottom layer consists only of input units, while the top layer consists of output units. There may be more than one layer of *hidden* units. When a training pattern is presented to the input layer, the inputs propagate forward and a non-input unit combines all its inputs by means of a logistic activation function to determine its own activation value. In BP training algorithm, the weights are updated during a backward sweep of the net from the output units down to the inputs. See [7] for further details.

In the work reported here, the standard BP is modified and parallelized in six different ways, namely:

1. Parallel Standard Back Propagation (PSBP),
2. Parallel Pipelining Back Propagation (PPBP),
3. Parallel Averaging Back Propagation (PABP),
4. Parallel Standard Steepest Descent (PSSD),
5. Parallel Pipelining Steepest Descent (PPSD),
6. Parallel Averaging Steepest Descent (PASD).

Loosely speaking, these six algorithms may be divided into two categories: the first three follow the BP approach in updating connection weights of the net after each pattern presentation, while the other three take the standard Steepest Descent approach and update the weights after each epoch (i.e. after all the patterns have been presented once).

In PSBP, each node learns its own patterns, and updates its own image of the net after each pattern presentation as in normal BP. At the end of an epoch, the nodes send their own total change of each weight during that epoch to each other by Alternate Direction Exchange (ADE) and update the net according to the generalized delta rule.

In PPBP, a node is chosen as the Manager, and an epoch is divided into pf (= *pipelining factor*) rounds. Each node subdivides the learning patterns it receives so as to learn as evenly as possible on each round. In each round, the Manager updates the connection weights from the computation results of the other nodes based on the old weights of the previous round, broadcasts the new weights to all nodes, and waits for the new computation results from all other nodes. Simultaneously, each node works on its patterns of that round with the old weights of the net using BP, and sends its own total change of each weight to the Manager using Hypercube Gathering (HG).

In PABP, each node learns its own patterns, and updates its own image of the net after each pattern presentation as in BP. At the end of an epoch, however, each node sends its own total change of each weight during that epoch as well as its weights to all nearest neighbors, while receiving theirs. It updates each weight by averaging the original values of the corresponding weights collected from all neighbors (i.e. those values at the beginning of that epoch), and then adding to the average all the corresponding weight changes.

The remaining three algorithms PSSD, PPSD, and PASD are Steepest Descent analogues of PSBP, PPBP, and PABP, respectively. Thus in these cases the weights are updated only after the whole set of input-output patterns have been processed.

Training patterns are assigned to the processors as evenly as possible in all cases. The hypercube gathering communication mechanism used by each method is summarized in Table 1. Refer to [2, 5] for details.

We note that the communication scheme also affects the locality of information. By using ADE, every node shares its information with each other and has a global view of the network. In HG, a node 'knows' the information that belongs to those children on its subtree of the hypercube spanning tree. The higher a node is in the tree, the more it knows; and the root knows all. By Nearest Neighbor Communication (NNC) a node only 'knows' information about its nearest neighbors. Intuitively,

[1]Research partially supported by UCSB Micro Proposal No. 615287; Agency Award No. 583002 (ESL Incorporated).

an algorithm using simpler communication such as NNC saves time spent on communication, but it may take more learning epochs as the nodes do not have the global image of the whole net. This is not the case if a more sophisticated communication method is adopted.

Table 1: Hypercube Gathering Schemes for Different Methods

METHOD	GATHERING SCHEME
Standard	Alternate Direction Exchange
Pipelining	Hypercube Gathering
Averaging	Nearest Neighbor Communication

Detailed description of a number of other communication methods and topological properties of hypercubes can be found in [9], [8], and [1].

Note that except PPSD and PPBP in which one node is chosen as the Manager to update the neural net, each node in all other algorithms executes the same set of operations. Also except PSSD, other algorithms will behave differently with different number of nodes used. For BP variations (PSBP, PPBP, and PABP), the weights are updated after each pattern presentation and different number of nodes will result in different distributions of learning patterns. For PABP and PASD, different number of nodes means different number of neighbors, and this will affect the amount of information a node may acquire during each learning epoch. Note also that some degree of relaxation is embedded in the averaging method as each processor just updates its knowledge based on the information it receives from those available locally around it.

3 Algorithm Performance Evaluation Tests

Two medium-size problems are used to evaluate the algorithms, using PDP models that consist of a single hidden layer with varying number of hidden units: vowel classification, and speech-to-text conversion (based on the NETtalk experiment).

These problems were chosen as BP is found to perform quite well on speech recognition [3, 6, 10]. The two problems have very different characteristics as summarized in Table 2, and the algorithms are found to behave differently on these two problems.

3.1 Vowel Classification Test

In this experiment a neural net is trained using BP to classify 9 English steady state vowels[2] in an /h-d/ context [4] uttered by 5 male speakers.

3.1.1 Test Details

- Number of learning patterns: 225.
- Number of input units: 17.
- Number of of output units: 9.
- Size of net with 4 hidden units: $4 \times (17 + 9) = 104$.
- Weights are initialized to random real numbers in the range 0.0 to 1.0.
- Terminating criterion: 90% correctness in maximum 10000 epochs.
- Message size: 5.168*Kbytes* for each of PSBP, PSSD, PPBP and PPSD, and 10.336*Kbytes* for PABP and PASD.

3.1.2 Test Results and Observations

Optimal results for vowel classification test on a 4-cube appear in Table 3 and the efficiency results are given in Table 4. Denoting by T the processing time, by N the number of epochs, and by pf the pipelining factor used, some observations are as follows:

A. With regard to the optimal results of vowels classification test on a 4-cube:

   We find that each of the standard and pipelining methods requires similar number of epochs to complete learning, regardless of which of the two approaches (SD or BP) is used. PASD works quite well and is comparable to PSBP. In terms of the number of training epochs required, pipelining method learns faster with a larger pipelining factor. The performance of the various methods on T/N ratio is in the following ascending order:

   standard < pipelining(with appropriate pf) < averaging.

B. With regard to the efficiency results:

   In general, efficiency decreases with the dimension of hypercube used. More precisely, the efficiency on high dimensional cubes decreases in the following order:

   pipelining ($pf = 2$) > standard > pipelining ($pf = 3$) > pipelining ($pf = 4$) > averaging.

   The reason is that pipelining method with small pf may overlap communication time with computation time, while that with large pf does not and requires more communication traffic. In spite of the simplicity of its communication scheme, averaging method has to exchange more information, and thus spends more time in communication. However, some amount of time may have been saved because of its asynchronous nature. The test results show that BP variations have higher efficiency than the SD variations.

3.2 Text-to-Speech Conversion Test

This experiment is based on the NETtalk experiment of [11] in which a neural net is trained using BP to pronounce English text, i.e., to convert a string of letters to a string of phonemes. Through the efforts of T.J. Sejnowski and others, a corpus of 20008 words was created for the NETtalk experiment. A small subset of 100 words, randomly selected from the Corpus and concatenated together to form a continuous text is used to evaluate the performance of the algorithms. The test procedure is adopted from [11]. Each letter is represented *locally* by 27 binary inputs, and *distributed* representation is employed in encoding phonemes by 26 binary outputs.

3.2.1 Test Details

- Number of learning patterns: 828 (100 words with inter-word spacing, giving 834 characters).
- Number of input units: 189 (27 inputs for each letter in the 7-letter window).
- Number of output units: 26 (21 for articulatory features plus 5 for stress and boundary).
- Size of net with 30 hidden units: $30 \times (189 + 26) = 6450$.
- Weights are initialized to random real numbers in the range of -0.3 to 0.3.
- Terminating criterion: 90% correctness in maximum 30 epochs.
- Message size: 52.496*Kbytes* for each of PSBP, PSSD, PPBP and PPSD, and 104.544 *Kbytes* for PABP and PASD.
- Measure of correctness: an output bit is said to be learned if the actual output deviates from the desired one by an amount less that or equal to 0.2.

Comparisons are done on a 'likely' optimal setting under which maximum exact matches are attained with 30 hidden units at the end of 30 epochs.

3.2.2 Test Results and Observations

Optimal results for text-to-speech conversion test appear in Table 5 while the effect of cube size on a net with 30 hidden units is given in Graph 1. Some observations are as follows:

A. With regard to the optimal results of text-to-speech conversion test on a 4-cube:

   We find that only the BP variations work! All the SD variations reach local minima after few learning epochs. All BP variations used almost the same amount of time to complete an epoch. Furthermore BP variations perform best (in terms of percentage of correctness) with the least number of processors; that is, 1 for PSBP or PABP, and 2 for PPBP.

[2] Data came from the B subset of the Speech Technology Laboratory Vowel Database, courtesy of Speech Technology Laboratory - a division of the Panasonic Technologies Inc. of Matsushita in Santa Barbara.

B. With regard to the effect of hypercube dimension:
For all BP variations, performance deteriorates with increasing hypercube dimension. PABP outperforms PSBP in cubes of dimensions greater than 1. PPBP performs better with increasing pipelining factor until it reaches ≈ 13.

C. With regard to the effect of continuous learning:

For PABP, the net continues to learn as more patterns are presented as shown in Graph 2. This is quite surprising considering that PSBP in 5-cube does not work at all, while PABP in 5-cube reaches $\approx$60% correctness after 90 training epochs, and PABP in 2-cube passes the 90% correctness mark. Whereas for PPBP, Graph 3 shows that performance initially increases with more presentations and saturates after $\approx$40 epochs for $pf = 13$, and $\approx$50 epochs for $pf = 6$. Also neither achieves the 90% correctness mark in both cases. Although within 30 training epochs most methods applied on large cube does not reach 90% correctness, some of them do give good results in relatively short time.

4 Conclusions

The two tested problems behave very differently under the various parallelized BP methods. The evaluation of the methods is difficult because of the many parameters involved. The vowel recognition has no preference to the approaches SD or BP that the learning schemes adopt. It gives similar results with all of the six parallelized BP methods. On the other hand, SD approach fails in speech-to-text conversion. The conversion seems to favor relaxation, requires more prompt incorporation of past

Table 2: Comparison of Characteristics of the Two Experiments

COMPARISON	VOWEL CLASS.	TEXT-TO-SPEECH CONVERSION
Inherent Parallelism	−	+++
Number of Patterns	−	+
Net Size	−	++
No. of Epochs Needed	++	−
Process Time/Epoch	−	++

(− means less while + means more)

experience (learned knowledge), and prefers generalization of knowledge combined from a smaller but reasonably rich set of 'dissimilar' patterns learned in parallel.

Of the three parallelized BP methods: standard, pipelining and averaging (SD or BP), pipelining method with an appropriate value of the pipelining factor learns better than the standard method. The value of the pipelining factor should be large enough to incorporate new knowledge to the neural net more promptly but yet small enough so that sufficient patterns are learned in order to derive novel knowledge. One of the drawbacks of the method is that it dedicates one of the processors to manage communication among the rest. The good performance of the averaging method is quite surprising. This method performs much better than the standard method in text-to-speech conversion in cubes of all sizes. Its performance on prolonged training with more pattern presentations is also better than the pipelining method, as the latter soon saturates.

A detailed exposition and analysis of the parallel variants of Back Propagation based training algorithms for speech related problems reported here is in preparation [2].

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[2] Ö. Eğecioğlu and B. K. Mak. Performance of parallelized back propagation algorithms for speech recognition and synthesis. In preparation, 1990.

[3] J. L. Elman and D. Zipser. Learning the hidden structure of speech. ICS Report 8701, Institute for Cognitive Science, University of California at San Diego, Feb. 1987.

[4] G. D. Haan, Ö. Eğecioğlu , and H. Wakita. Improving the performance of back propagation - trained vowel classifiers. *The Journal of the Acoustical Society of America*, 84, 1988. supplement 1, U4.

[5] B. K. Mak and Ö. Eğecioğlu . Communication parameter tests and parallel back propagation algorithms on iPSC/2 hypercube multiprocessor. *The Prcoeedings of the Fifth Distributed Memory Computing Conference*, 1990.

[6] S. M. Peeling, R.K. Moore, and M.J. Tomlinson. The multi-layer perceptron as a tool for speech pattern processing research. *Proc. IoA Autumn Conference on Speech and Hearing*, 1986.

[7] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1:Foundations. MIT Press, 1986.

[8] Y. Saad and M. H. Schultz. Data communication in hypercubes. Research Report YALEU/DCS/RR-428, Computer Science Department, Yale University, October 1985.

[9] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transaction on Computers*, 37:867-872, 1988.

[10] T. J. Sejnowski and C. R. Rosenberg. Nettalk: A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, John Hopkins University, 1986.

[11] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145-168, 1987.

Table 3: Optimal results of vowel classification test on a 4-cube
(pf = pipelining factor, H = number of hidden units,
η = learning rate, N = number of epochs required,
T = processing time, UP = number of patterns not learned)

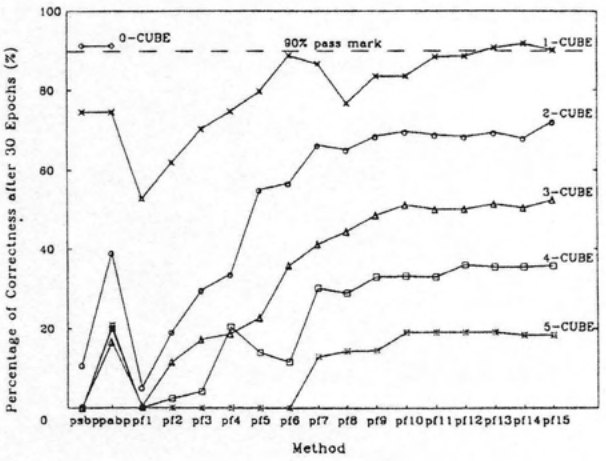
METHOD	OPTIMAL RESULT					
	H	η	N	$T(min)$	UP	T/N $(ms/epoch)$
PSSD	4	0.2	3568	7.60	21	128
PSBP	6	0.1	3482	13.7	18	236
PSBP†	4	0.1	5522	15.8	19	171
PPSD(pf=2)	4	0.3	4677	10.1	19	130
PPBP(pf=2)	4	0.2	4492	13.2	20	176
PPSD(pf=3)	6	0.2	2708	8.5	27	265
PPSD(pf=3)†	4	0.2	5426	12.0	24	144
PPBP(pf=3)	6	0.2	2213	9.6	22	259
PPBP(pf=3)†	4	0.2	4631	14.8	27	192
PPSD(pf=4)	4	0.5	2318	6.10	26	158
PPBP(pf=4)	4	0.4	1971	6.80	26	207
PASD	4	0.7	4873	12.1	16	148
PABP	4	0.8	9784	31.0	21	190

(† not optimal case but used for comparison)

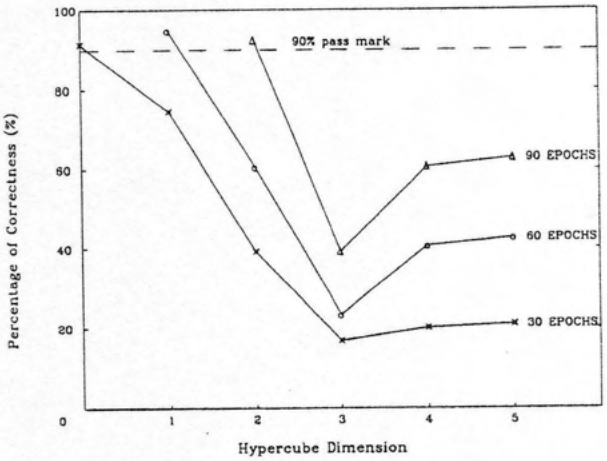
Table 4: Efficiency results of vowel classification test with optimal setting
(pf = pipelining factor, T_i = processing time for i processors,
E_i = efficiency for i processors, N = no. of epochs required)

METHOD	RESULT	0-CUBE	1-CUBE	2-CUBE	3-CUBE	4-CUBE	5-CUBE
PSSD	$T_i(min)$	80.4	41.2	21.2	12.6	7.60	4.90
	N	3243	3273	3249	3539	3568	3371
	$T_i/N(ms)$	1488	755	392	214	128	87.2
	E_i	100%	98.4%	95.2%	87.3%	73.4%	53.5%
PSBP	$T_i(min)$	168.1	115.7	56.3	23.7	13.7	8.30
	N	3317	5682	4284	3406	3482	3389
	$T_i/N(ms)$	3041	1539	789	417	236	147
	E_i	100%	98.8%	96.3%	91.1%	80.7%	64.5%
	$T_i(min)$		126.0	41.3	22.8	10.1	8.40
PPSD	N		5078	4846	5685	4677	5664
pf=2	$T_i/N(ms)$		1489	511	241	130	89.0
	E_i		100%	97.1%	88.3%	76.4%	54.0%
	$T_i(min)$		169.7		26.5	13.2	10.5
PPBP	N		4694		4670	4492	5502
pf=2	$T_i/N(ms)$		2169		340	176	115
	E_i		100%	—	91.1%	82.2%	60.8%
	$T_i(min)$		323.5	44.5	15.8	8.50	5.50
PPSD	N		9438	3770	2793	2708	2392
pf=3	$T_i/N(ms)$		2056	708	339	189	138
	E_i		100%	96.8%	86.6%	72.5%	48.1%
	$T_i(min)$		118.9	33.1	21.8	9.60	6.60
PPBP	N		2339	1910	2688	2213	2233
pf=3	$T_i/N(ms)$		3049	1041	487	259	177
	E_i		100%	97.6%	89.4%	78.5%	55.6%
	$T_i(min)$		61.7	18.5	9.30	6.10	4.50
PPSD	N		2471	2112	2123	2318	2095
pf=4	$T_i/N(ms)$		1498	526	263	158	129
	E_i		100%	94.9%	81.4%	63.2%	37.5%
	$T_i(min)$		105.5			6.80	6.00
PPBP	N		2903			1971	2340
pf=4	$T_i/N(ms)$		2181			207	154
	E_i		100%	—	—	70.2%	45.7%
PASD	$T_i(min)$					22.1	41.5
	N					4873	10000
	$T_i/N(ms)$					300	249
	E_i	—	—	—	—	—	—
PABP	$T_i(min)$	51.9	28.9	27.4		60.4	
	N	1434	1564	2661		9784	
	$T_i/N(ms)$	2172	1109	618		370	
	E_i	100%	98.0%	87.8%	—	36.8%	—

(— indicates unsuccessful learning)



Graph 1: Effect of cube size on text-to-speech conversion with H=30

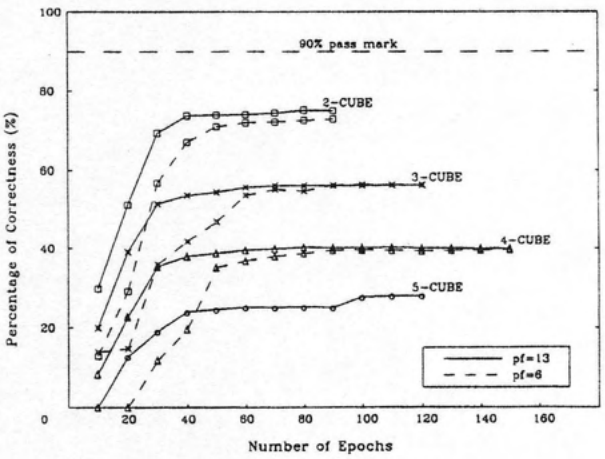


Graph 2: Detail results of PABP on text-to-speech conversion

Table 5: Optimal results of text-to-speech conversion test after 30 epochs
(pf = pipelining factor, H = number of hidden units,
P = number of processors required, η = learning rate,
T = processing time, PC = percentage of correctness)

METHOD	OPTIMAL RESULT, H = 30				
	P	η	$T(min)$	$PC(\%)$	$T/30(min/epoch)$
PSBP	1	1.0	161.7	91.4	5.39
PABP	1	1.0	161.7	91.4	5.39
PPBP(pf=1)	2	1.0	161.4	52.9	5.38
PPBP(pf=2)	2	0.9	166.1	62.0	5.54
PPBP(pf=3)	2	1.0	162.2	70.2	5.41
PPBP(pf=4)	2	1.0	162.1	74.6	5.40
PPBP(pf=5)	2	1.0	162.2	79.7	5.41
PPBP(pf=6)	2	1.0	162.5	88.8	5.42
PPBP(pf=7)	2	1.0	162.6	86.7	5.42

(Note: steepest descent variations do not work.)



Graph 3: Detail results of PPBP(pf=6,13) on text-to-speech conversion