A Multi-Computer Neural Network Architecture in a Virtual Sensor System Application

R.J.Howlett and S.D.Walters

Intelligent Signal Processing Laboratories, School of Engineering, University of Brighton, Moulsecoomb, Brighton, Sussex, BN2 4GJ, U.K.
Email: R.J.Howlett@brighton.ac.uk

Keywords: neural network, parallel processing, virtual sensors.

Abstract

The multi-layer perceptron (MLP) neural network is recognised to have a convergence rate which is slower than desired. Multi-computer systems are attractive platforms for the implementation of neural networks, offering the potential for achieving faster convergence through increased processing power. However, multi-computer implementations of the MLP that are found in the literature offer an improvement in performance which is less than would be anticipated due to the inherent high communications overhead. This paper describes the Class-Distributed (C-D) Neural Network, a method of implementing a modified feed-forward neural network using the back-propagation learning algorithm on a multi-computer system to form a distributed neural network classifier. The inter-processor communications requirement for the C-D network is minimal and the convergence rate is superior to that of comparable methods. The performance of the C-D network is evaluated in an application where the slow convergence rate of the MLP has been found to be a problem. The application involves a virtual sensor system in an internal combustion engine. The virtual sensor provides an indirect estimate of the air-fuel ratio in the engine. The C-D network implementation of the virtual air-fuel ratio sensor offers a convergence rate superior to a comparable MLP based system, and it demonstrates similar classification capabilities.

1. Introduction

The multi-layer perceptron (MLP) neural network, which uses the back-propagation learning algorithm, is widely employed as a classifier because of its good generalisation properties and compact internal structure. However, its convergence rate is poor, leading to extended training times. This can be problematic when the MLP is applied in practical situations, especially where the problem domain inherently involves a large and complex input space.

Multi-computer systems have appeared to be attractive platforms for the implementation of neural network algorithms, offering the potential for achieving faster convergence through increased processing power [1,2,3,4]. However, multi-computer implementations of the MLP that are found in the literature achieve an improvement in performance over single-processor versions that is not as good as would be anticipated. This owing to the high inter-processor communications overhead which is inherent in the parallel algorithms used.

2. Multi-Computer Implementations of the Multi-Layer Perceptron

Two methods of implementing an MLP network on a multi-computer system are described in the literature, *Training File Division* and *Data Decomposition.*

The Training File Division technique [1,2] involves placing a complete MLP network on each processor and dividing the training file equally among them. The delta-weight calculations can take place independently; however, to prevent weight incoherence, the weight corrections determined by each processor must be applied to the weights-set held on every processor. This can necessitate substantial inter-processor communication. The determination of a division of the training file that gives an optimum improvement in speed is difficult. In fact, it is impossible to predict, in a particular application, whether the method will achieve an improved convergence rate at all.

The Data Decomposition procedure, which is more commonly found in the literature [3,4], is to divide the neural network itself and place a portion of it on each processor. Neurons from the input-layer, hidden-layer and output-layer are equally divided and placed on each processor. The weights associated with the neurons which make up the network and the input and output vectors are distributed among the processors. These values form the data of the neural network program and as this data has been distributed or decomposed among the processors the method is an example of data-decomposition. The back-propagation algorithm is carried out on each processor in parallel, updating the weights on that processor. However, much of the data, for example output and weight values, is required globally and must be distributed to all the processors after each iteration. This represents a considerable communications overhead which is inevitable with this method. Consequently, the effects of communications saturation are apparent even with small numbers of processors and an improvement in speed is obtained which is less than optimum.

Different interconnection networks and topologies can be employed, however, it has been shown that the selection of the processor network topology does not strongly influence the communications overhead [4]. The effects of saturation are less discernible with larger networks because the communications overhead is lower in comparison with the computational load; smaller networks do not scale well [3].

Thus, the data-decomposition procedure leads to a requirement for a high communications bandwidth which low-cost platforms are unable to satisfy. The experimental work described in the next section of the paper illustrates this.

3. Experimental Evaluation of the Data-Decomposition Method

An MLP neural network was implemented using a multi-computer system that could be configured to use 1, 2, 4 or 8 processors. The Inmos (SGS-Thomson) Transputer, that has featured in the literature as a low-cost platform for concurrent algorithm development [5], was used to execute the distributed network. A network having 50 inputs and 8 outputs was used and experiments were conducted using various numbers of neurons in the hidden layer.

Figure 1 shows graphs of time per training iteration for 8, 16, 24 and 48 hidden nodes. The graphs indicate that as the number of processors is increased, there is progressively less improvement obtained in terms of reduction in iteration time. This is because the communication overhead increases with the number of processors.

Figure 2 shows the results displayed in a different way. The performance of a multi-computer system is often expressed in terms of the *speed-up*, the ratio between the speed of a multi-computer system and that of a single processor, where both perform the same task.

Thus,

$$\textit{speed-up} = t_s / t_p \qquad (1)$$

where

t_s = time taken by a single processor

t_p = time taken by multi-computer system

with the same algorithm under execution in both cases. In a system where the time taken for communication between the processors was zero, the speed-up value would be equal to the number of processors.

Figure 2 shows that the speed-up obtained for a given number of processors is greater for the larger network (the network with the greater number of hidden neurons). In the case of the smallest network, that which has eight hidden neurons, there is a reduction in speed-up (rather than an increase) when the number of processors is increased from four to eight.

It is acknowledged that the Transputer is an elderly processor. Multiple-Transputer systems still form useful platforms for algorithm development, however, the Transputer is unlikely to be used in new practical applications. The more recent Texas digital signal processor (DSP) devices are similarly attractive as low-cost parallel processing platforms, being equipped with between four and six high speed communications links which facilitate connection into processor arrays. The TMS320C40 has a peak MFLOP rate which is higher than that of the Transputer (80MFLOPS for a TMS320C40 with a 60MHz clock compared with 4.3 MFLOPS for a 30MHz IMST805) [6,7]. However, the communications bandwidth has not increased in proportion (28Mbyte/s for the TMS320C40 and 2.4Mbyte/s for the IMST805). Thus, the onset of communications saturation would be expected to occur even earlier with these newer devices. Low-cost parallel processing systems can also be constructed from single-card PC-compatible processing modules which use the Intel Pentium II or III processor. Modules with clock speeds of several hundred MHz are available at modest cost, however, the inter-processor communications channels that are available have only a very restricted bandwidth.

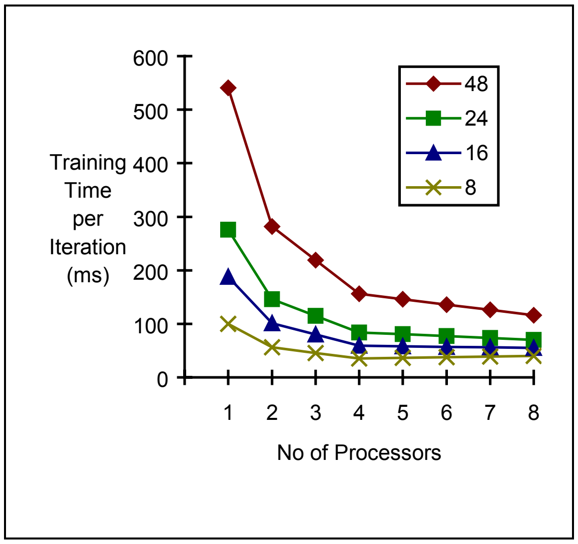

Figure 1: Training time per iteration for Data-Decomposed MLP : 8, 16, 24 & 48 hidden nodes

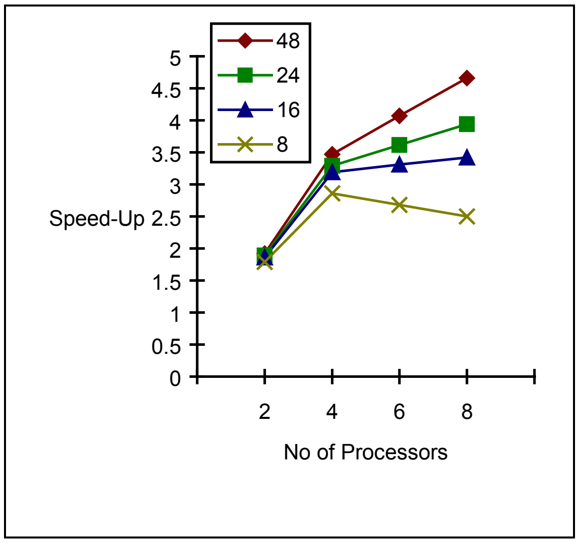

Figure 2: Speed-up factor for Data-Decomposed MLP : 8,16,24 & 48 hidden nodes

Neural network training algorithms which efficiently utilise the available low-cost platforms must make small demands on the inter-processor communications bandwidth. Data decomposition methods do not achieve this and so other algorithms are required that are characterised by an inherently low communications requirement.

4. The Class-Distributed Neural Network

The Class-Distributed (C-D) neural network is a fast-learning pattern-classifier based on a modified back-propagation gradient-descent algorithm efficiently executing on a distributed architecture. The inter-processor communications requirement for this new network is minimal, making efficient execution possible on low-cost parallel processing systems that have a restricted communications bandwidth [8].

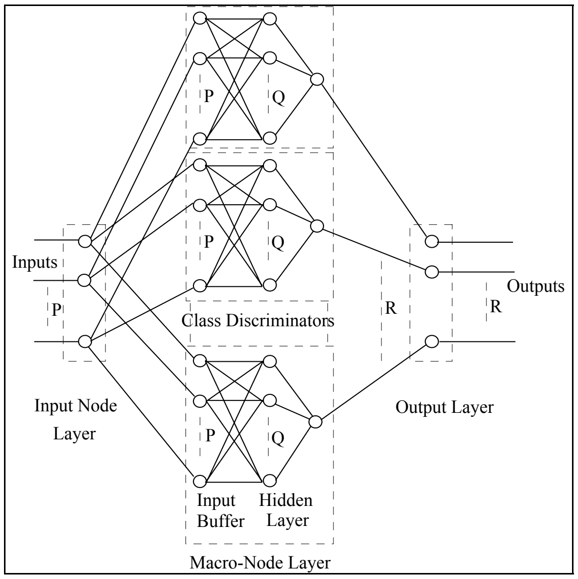

Figure 3 The Class-Distributed Neural Network Architecture

The C-D neural network architecture is illustrated in Figure 3. The input layer consists of buffer neurons, which may perform scaling and distributes copies of the input vectors to the class discriminators in the category layer. The output layer implements a simple winner-takes-all paradigm during recall. The class-discriminators in the category layer function as complex feature detectors, assigning input patterns to a class. The class-discriminators are used as the basis for dividing the functional capabilities of the network among the processors of the multi-computer system. Thus, the network is decomposed by class and for this reason it is termed a *Class-Distributed* network.

The input node layer passes a *p*-dimensional feature vector *x*, which is to be assigned to one of *r* different classes, to the category layer. The class-discriminators in the category layer are macro-nodes, each consisting of a feed-forward network having an input buffer, a hidden layer of active (weight-carrying) neurons, and a single active output neuron. Let there be a training file $F = \{S_1 \cup S_2 \cup ... \cup S_r\}$ where S_j is a subset of *F* containing all training vectors belonging to class *j*. During the training phase the macro-nodes perform total gradient descent independently in parallel, each concurrently executing a modified cumulative back-propagation algorithm. Each macro-node receives the same input vector during each delta-weight calculation, but a different element of the desired output vector. Each macro-node trains towards a different convergence criterion; for example, at the termination of the learning phase, the output of the j_{th} class discriminator is $y_j \geq 1 - T_c \ \forall \ x_n \in S_j$ and $y_j \leq T_c \ \forall \ x_m \in S_i \ (i = 1, ... r) \ i \neq j$ where T_c is a convergence threshold.

During recall the same input vector is applied to all the macro-node input buffers, via the input layer. The forward pass calculations are performed and the output value of each macro-node is passed to the output layer. The output of only one macro-node becomes active, that macro-node being the one associated with the input pattern. The winner-takes-all layer arbitrates between the output values of the macro-nodes.

Each macro-node in the category layer forms a sufficiently complex functional block to make it viable as a discrete software process or task. The C-D network is executed on a multi-computer platform by implementing each class-discriminator as a process and distributing these processes among the processors. The distribution of processes need not be the same during training and recall. The computational requirements of training will often demand that each class-discriminator be allocated its own processor. However, during recall, the application could be sufficiently undemanding for a high enough categorisation rate to be achieved by all class-discriminators executing on a single processor.

The inter-processor communications requirement consists merely of the distribution of the training vectors and collation of the output values. As this is negligible, wide communications bandwidth is not required and the choice of inter-connection topology does not influence performance.

The software design of the C-D network adheres to the requirements of modern structured software design methodologies more closely than the data-decomposition approach. The network is effectively object-oriented, achieving functional-encapsulation and data-encapsulation and so obeying the principles of these methodologies. The C-D network exhibits benefits not apparent with the data-decomposed back-propagation method, for example, high logical cohesion, low algorithmic complexity leading to ease of verification and validation, and good maintainability properties. Error elimination is facilitated because each class-discriminator can be coded, verified and validated independently. The program may be coded and tested on a single processor, and this code may then be duplicated with only minor modifications and placed on the multi-processor nodes.

5. A Neural Network Virtual Sensor

Two of the most important considerations in the area of motor-vehicle engine design are fuel economy and the reduction of harmful exhaust emissions. Europe, the United States, and much of the rest of the world, has legislative controls which govern the permissible levels of pollutants in the exhaust of Internal Combustion (IC) engines [9]. Maintaining these standards in current engines demands strict control of operational parameters using microprocessor-based Engine Management Systems (EMS). A parameter that is of considerable importance in determining the operating point of the engine, its output power and emission levels, is the air-fuel ratio (AFR). The air-fuel ratio is often defined in terms of the *excess air factor*, or *lambda* ratio:-

$$\text{lambda} = AFR / AFR_{st} \qquad (1)$$

where

AFR = the current air fuel ratio

and

AFR_{st} = the *stoichiometric* air fuel ratio

Lambda is defined such that a lambda-ratio of unity corresponds to an air-fuel ratio of approximately 14.7:1 at normal temperature and pressure, when the fuel is petrol or gasoline. This is termed the *stoichiometric ratio*, and corresponds to the proportions by mass of air and fuel that are required for complete combustion. A greater proportion of fuel gives a lambda-ratio of less than unity, termed a *rich* mixture, while a greater proportion of air gives a lambda-ratio of greater than unity, termed a *weak* or *lean* mixture. Maximum power is obtained when the lambda-ratio is approximately 0.9 and minimum fuel consumption occurs when the lambda-ratio is approximately 1.1.

Current vehicles reduce emission levels to within legislative limits by converting the exhaust gases into less toxic products using three-way catalytic converters. For optimum effect, three-way catalytic converters require that the lambda-ratio is closely maintained at stoichiometric (unity). In modern engines, a *lambda-sensor,* mounted in the exhaust stream, determines whether the lambda is above or below unity from the amount of oxygen present. The EMS uses this to adjust the fuel pulse width to keep the lambda-ratio approximately at unity. Power units currently under development, for example the gasoline direct injection (GDI) engine, may involve operation in lean-of-stoichiometric regions of the characteristics of the engine. Precise control of the air-fuel ratio is of considerable importance here also.

The lambda-sensor that is installed in most production vehicles has a steep voltage-lambda characteristic. It is used to indicate whether the value of lambda is above or below unity, but it is unable to provide an accurate analogue measurement of air-fuel ratio. Accurate measurements can be made using what are referred to as *wideband* lambda-sensors, but these are generally too expensive for use in production engines. The hardware lambda sensor represents an undesirable cost burden, and its elimination, or replacement by a lower-cost technique, would be advantageous.

The C-D network was evaluated as part of a *virtual sensor* system which estimated the lambda-ratio using a technique known as Spark Voltage Characterisation (SVC). The SVC method involves the analysis of the time-varying voltage that appears across the spark-plug, due to the ignition system, for monitoring combustion phenomena in the cylinder, and estimating quantities such as the lambda-ratio [10, 11, 12]. The SVC method exhibits a number of advantages over the conventional lambda sensor: the hardware sensor is not required, providing a cost saving, and it has the potential for wideband lambda measurement.

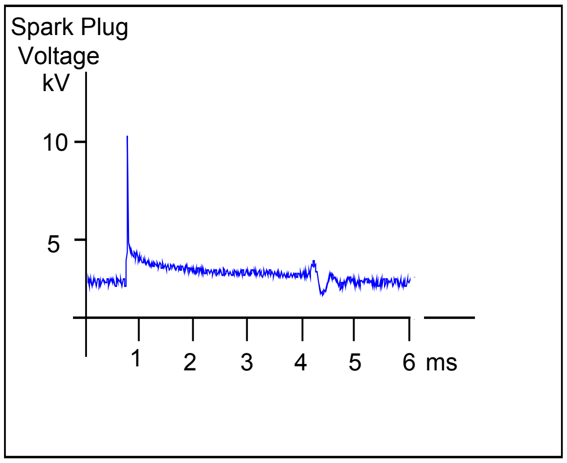

Figure 4: Typical spark-plug voltage waveform

Figure 4 shows the spark-plug voltage waveform obtained from a typical motor-vehicle engine ignition system. The spark-plug voltage waveform possesses a number of predictable phases. As the high-voltage pulse is generated by the ignition-system, the voltage across the spark-gap rises to between approximately six and 22 kV, before breakdown occurs. Breakdown is accompanied by a decrease in voltage, giving a characteristic voltage spike of approximately 10μs in duration. This is followed by a glow-discharge region of a few milliseconds duration, that appears as the tail of the waveform.

Changes in the value of the lambda-ratio would be expected to influence both the maximum value of the breakdown voltage and the time-varying voltage characteristic of the arc and glow discharge phases. A formal relationship between the value of lambda-ratio and the instantaneous voltage at any particular point on the spark-voltage characteristic, and the other variables, is not easy to formulate and may not exist. However, theoretical analysis predicts a possible correlation between the vector formed by periodic sampling of the voltage at the spark-plug over the spark time, termed the *spark-voltage vector*, and the lambda-ratio. With suitable pre-processing and training, a neural-network is a suitable tool for estimating the lambda-ratio by associating it with the spark voltage vector.

A problem that is encountered when an MLP neural network is used to carry out SVC analysis is that a large amount of training data is necessary. The consequence of this is a convergence time during training which can be impractically long [10, 11]. Neural network algorithms that naturally converge more rapidly than the monolithic MLP would offer advantages in this context.

6. Experimental Evaluation of the C-D Neural Network

Experiments are described here where the aim was to compare the convergence time and accuracy of the C-D network with that of a monolithic MLP. A practical application domain was used for the comparison, namely, an air-fuel ratio virtual sensor using the SVC method.

The experimental work was conducted using a single-cylinder 98.2cc capacity four-stroke engine. The time varying voltage at the spark plug was recorded for values of lambda-ratio of stoichiometric and stoichiometric $\pm 10\%$ at a fixed value of engine speed and load torque.

Training files were constructed from 250 spark voltage records corresponding to each lambda value, pre-processed and compressed into 12-element input vectors each associated with a 3-element output vector. The fuel pulse width and dynamometer were adjusted to give an engine speed and a lambda-ratio of the desired values. Instantaneous spark-voltage vectors of the form $V_n = (v_1, v_2 .. v_p)$ were created by recording the voltage at the spark-plug at measured intervals of time. A more detailed description of the data-capture system is published elsewhere [12]. Each spark-voltage vector was associated with a desired-output vector, $D_r = (0,0,1)$, $D_c = (0,1,0)$ and $D_w = (1,0,0)$, depending on whether the lambda-value, measured by the exhaust gas analyser, was rich, correct or weak, respectively. Three sets of spark-voltage vectors and their associated desired-output vectors were obtained, S_r, S_c and S_w, corresponding to rich, correct and weak lambda values. These vectors were combined into a single training-file, $F = \{S_r \cup S_c \cup S_w\}$.

Similar files, having the same construction, but using data that was not used for training, were created for test purposes.

The execution platform was based on 166MHz Intel Pentium II processor nodes connected by a low-bandwidth (approximately 1.0 Mbyte/s) network.

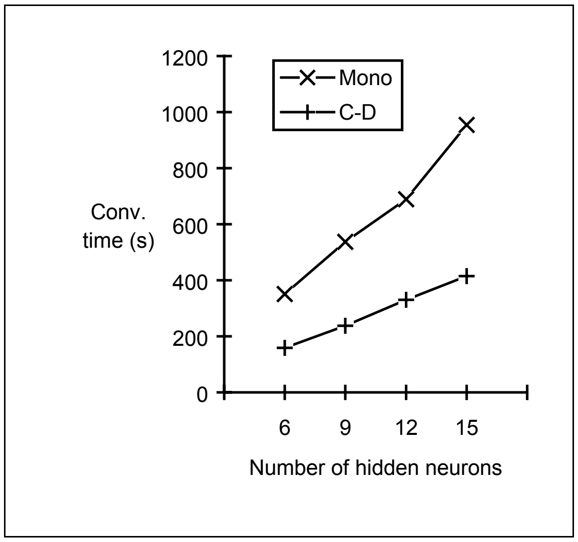

Figure 5: Convergence times for Monolithic and C-D networks

Figure 5 shows convergence times for the C-D network compared with that of the monolithic network for different numbers of hidden nodes when the above training files were used. The iteration time and time to convergence of each class-discriminator is considerably lower than for a comparable MLP network executing on a serial processor (termed here a *monolithic* network). The weights in both types of network are adjusted by adding to them a *delta-weight* proportional to the neuron error. In a monolithic network, the error in element q of the hidden layer is related to the output X_q, the weight between the hidden and output layers, W_{qr}, and the output layer error E_r, by:-

$$E_q[hid] = X_q[hid](1 - X_q[hid]) \sum_{r=0}^{R} (E_r[out] W_{qr}[out]) \quad (1)$$

In the C-D network, however, the single output node results in the less computationally onerous procedure

$$E_q[hid] = X_q[hid](1 - X_q[hid])(E[out] W_r[out]) \quad (2)$$

resulting in a lower per-iteration time during training than a monolithic network. This is illustrated by Figure 6, which shows that the epoch time for each macro-node of the C-D network was considerably lower than that of the monolithic network.

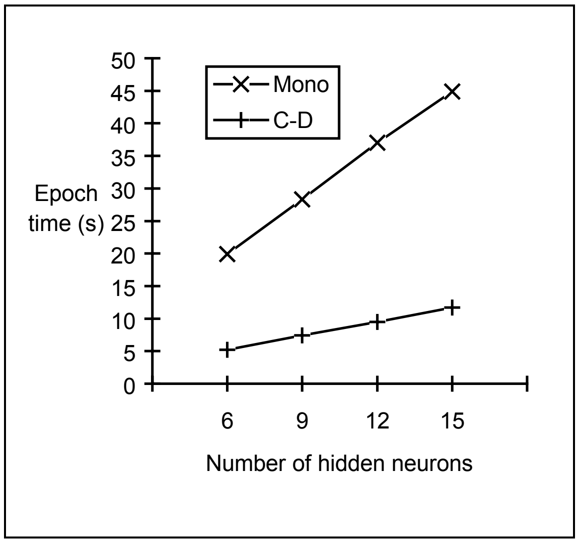

Figure 6: Epoch time for Various Numbers of Hidden Nodes

A second reason for the improved convergence rate of the C-D network is that the number of hidden neurons required by each class-discriminator is often less than that needed by the monolithic network. This parameter is dependant on the problem domain and the topology of the input space [13], but the number of hyper-planes needed to define a single region in input space is in many cases fewer than the number needed to separate multiple regions. Further, the number of weight adjustments needed for the network to converge from its random start condition in weight space, to the desired convergence condition, is statistically likely to be lower where there are fewer weights. For similar reasons, the C-D network achieves faster recall than an equivalent monolithic network.

No Hidden Nodes	6	9	12	15
Speed-gain	2.20	2.26	2.28	2.3

Table 1: Speed gain for various numbers of hidden nodes

The *speed-gain* is defined as the ratio of the times-to-convergence of the monolithic and C-D networks, $g_m = t_m / t_{cd}$. Table 1 shows that in the case of this application, g_m increases marginally with network size. Table 2 illustrates the comparative abilities of the two networks to correctly determine the lambda ratio from a test spark voltage vector. The table shows that these is little appreciable difference in the classification capabilities of the two networks. Thus, the C-D network achieves a clear speed improvement over the monolithic network during training with no appreciable classification penalty.

No Hidden Nodes	6	9	12	15
Monolithic	97.8	97.6	98.2	97.6
C-D	97.8	97.3	98.0	97.3

Table 2: Classification rates for Monolithic and C-D networks

If the monolithic network had been decomposed onto the multi-computer system by the data-decomposition method, instead of by class-distribution, the speed improvement which would have been obtained is the *speed-up*, $g_s = t_m / t_{dd}$, the ratio of times-to-convergence of the monolithic network and the data-decomposed network. The maximum theoretical value of speed-up obtainable with three processors is 3.0. However, previous studies have shown that with this size of network and the available communications bandwidth, communications saturation is likely to reduce the practically achievable speed-up to unity or less when the data decomposition method is used. The network (in terms of the number of weights) was comparable with the smallest for which results were displayed in Figures 1 and 2. No improvement in convergence rate would have been likely, in fact the convergence rate would be likely to be worse for the data-decomposed MLP than for the monolithic MLP. Given these constraints, the comparative speed improvement which was achieved by the C-D network is very attractive.

7. Conclusion

The C-D neural network architecture offers faster training and recall when executed on a multi-computer platform compared to a monolithic MLP. The inter-processor communications requirement is negligible, permitting efficient execution on a parallel processing system with a low communications bandwidth. For this reason, the convergence rate of the C-D network is superior to that of a data-decomposed MLP when the communications bandwidth is limited. In the example application, the IC engine virtual lambda sensor, the C-D network offered an improvement in convergence rate, and would be expected to provide an improvement in operational speed in recall, although this was not measured. Use of a data-decomposed MLP in this application would not have been feasible as the multi-computer execution platform possessed a very low bandwidth inter-communications network.

References

[1] Parker, K.L. and Thornbrugh, A.L. Parallelized back-propagation training and its effectiveness. *International Joint Conference on Neural Networks.* Vol.2. pp.179-182. Washington, DC. 1990.

[2] Paugam-Moisy, H. On a parallel algorithm for back-propagation by partitioning the training set. *Neural Networks and their Applications, 5th International Conference - Neuro-Nimes*. pp.53-65. France. 1992.

[3] Yamazaki, K., Kanatani, T., Wanatanabe, T., and Tokumaru, H. Parallelization of back-propagation learning using several interconnection networks on a multi-Transputer system. In *Transputer Applications and Systems*. Eds. Grebe, R. et al. pp.668-680. IOS Press. Amsterdam. 1993.

[4] Murre, J.M.J. Transputers and neural networks: An analysis of implementation constraints and performance. *IEEE Transactions on Neural Networks*. Vol.4. No.2. pp.284-292. 1993.

[5] McLoone, S. And Irwin, G.W. Fast parallel off-line training of multi-layer perceptrons. *IEEE Transactions on Neural Networks.* Vol.8. No.3. pp. 646-653. 1997.

[6] Graham, I. and King, T. *The Transputer Handbook.* Prentice Hall. p.13. 1990.

[7] Texas Instruments. *TMS320C4X User's Guide*. 1996.

[8] Howlett, R.J. A distributed neural network for machine vision. PhD Thesis. University of Brighton. 1995.

[9] Cardini, P. Focus on emissions: Going for the burn. *Automotive Engineer.* Vol. 24. No. 8. pp. 48-52. 1999.

[10] Howlett, R.J. Monitoring and control of an internal combustion engine using neural and fuzzy techniques. *International Conference on the Engineering of Intelligent Systems, EIS'98.* Tenerife, Spain. Feb, 1998.

[11] Howlett, R.J., Walters, S.D., Howson, P.A. and Park, I. Air-fuel ratio measurement in an internal combustion engine using a neural network. *International Conference on Advances in Vehicle Control and Safety, AVCS'98.* Amiens, France. July, 1998.

[12] Howlett, R.J., de Zoysa, M.M., Walters, S.D. and Howson, P.A. Neural network techniques for monitoring and control of internal combustion engines. *International ICSC Symposium on Intelligent Industrial Automation.* Genoa, Italy. June, 1999.

[13] Hush, D.R., Horne, B. and Salas, J.M. Error surfaces for multilayer perceptrons. *IEEE Trans. on Systems, Man and Cybernetics*. Vol.22. No.5. pp.1152-1161. 1992.

First submitted 30 November 1999 by Dr R.J.Howlett
This revised version submitted 10 March 2000