Distributed Computing Methodology for Training Neural Networks in an Image–guided Diagnostic Application

V.P. Plagianakos [a,c], G.D. Magoulas [b], M.N. Vrahatis [a,c,*]

[a] *Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece*

[b] *School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, United Kingdom*

[c] *University of Patras Artificial Intelligence Research Center–UPAIRC*

Abstract

Distributed Computing is a process through which a set of computers connected by a network is used collectively to solve a single problem. In this paper, we propose a distributed computing methodology for training neural networks for the detection of lesions in colonoscopy. Our approach is based on partitioning the training set across multiple processors using a Parallel Virtual Machine. In this way, interconnected computers of varied architectures can be used for the distributed evaluation of the error function and gradient values, and, thus, training neural networks utilizing various learning methods. The proposed methodology has large granularity and low synchronization, and has been implemented and tested. Our results indicate that the Parallel Virtual Machine implementation of the training algorithms developed leads to considerable speedup, especially when large network architectures and training sets are used.

Key words: Distributed computing, parallel implementations, parallel virtual machine–PVM, backpropagation training, image–guided diagnosis and surgery

[*] Corresponding author
Email addresses: vpp@math.upatras.gr (V.P. Plagianakos), gmagoulas@dcs.bbk.ac.uk (G.D. Magoulas), vrahatis@math.upatras.gr (M.N. Vrahatis).

1 Introduction

Distributed systems allow the deployment and utilization of heterogeneous, network–connected computing resources and offer the potential to analyze, share and manage medical imaging information in more flexible and intelligent ways, with a view to making evidence–based decisions, recognizing patterns and generating new hypotheses on–line [1]. The emergence of grid protocols in conjunction with distributed computing offers CPU and data handling capabilities to users and could provide decision support in clinical diagnosis [2]. Minimally invasive, image–guided diagnostic procedures and surgery are particularly benefited from advanced software and hardware infrastructures [3,4]. In this context, the integration of navigation systems with high tracking accuracy and manoeuvrability, real–time services, such as analysis of imaging data and classification, parallelization of computational methods, detection of similarities with data stored in collaborating sites, comparison of patient's images against the norm, information sharing, and e–collaboration with other experts would definitely increase the efficiency of typical diagnostic procedures and surgeries [3–8].

Towards this direction, this paper investigates the use of a distributed computing methodology for an image–guided diagnostic scheme that employs Multi–Layer Perceptrons (MLPs) for the detection of lesions in colonoscopy images and video sequences. To this end, we propose a way to partition the training set across multiple processors and we evaluate the speed performance of the distributed scheme with respect to a single processor implementation. The proposed distributed computing methodology utilizes the Parallel Virtual Machine–PVM [9–11] software tools and libraries.

2 Background

In medical practice, minimally invasive techniques, such as computed tomography, ultrasonography, confocal microscopy, computed radiography, magnetic resonance imaging, or endoscopy are now permitting visualization of previously inaccessible regions of the body. Their objective is to increase expert's ability in identifying malignant regions and decrease the need for intervention while maintaining the ability for accurate diagnosis. Furthermore, it is possible to examine a larger area, study living tissue in vivo – possibly at a distance [12] – and, thus, minimize the shortcomings of biopsies, such as a limited number of tissue samples, a delay in diagnosis, infection, perforation, and discomfort for the patient.

Colorectal cancer is the second leading cause of cancer–related deaths in the

United States [13,14]. Screening is the current and most suitable prevention method for early detection and removal of colorectal polyps. If such polyps remain in the colon can possibly grow into malignant lesions. Colonoscopy is the most accurate screening technique for detecting polyps, also allowing biopsy of lesions and resection of most of the polyps [15]. Colonoscopic diagnosis is a particular challenging area, involving the extraction and interpretation of patterns from complex medical video sequences under variable perceptual conditions (resolution change, shadings, shadows, lighting condition variations, reflections etc.), hypothesis generation, and reasoning in relation to previous experiences of the medical experts [4,16–18]. When one considers that abnormalities are hard enough to diagnose, the problem is exaggerated greatly when the physicians do not know what they are looking for.

The use of intelligent approaches for the detection of lesions in colonoscopy has to meet a number of challenges [4,18]: the time varying nature of the process, changes in the perceptual direction of the physician, variations in the diffused light conditions. For example, though one can use bright lights, the effect in a tight organ is that light tends to diffuse which leads to some areas being clearly lit and others not so; thus potentially hiding abnormalities. Relating to diffused light and the restrictive nature of the organ, it is easily possible for shadows to appear, restricting further what is visible. Shadows can be caused by the endoscope itself, different sections and abnormalities themselves. Lastly, the limited manoeuvrability of the endoscope causes the views at which abnormalities are visible to be far from ideal; a bad view can easily exaggerate the noise present within the image and hide abnormalities.

In most of these cases, training examples or explicit knowledge are not able to capture all possible variations of the environment. Collaboration among experts, estimations of similarities with data held in remote sites, and fast analysis of imaging data can definitely increase the efficiency of the procedure.

3 Computational methods

Automatic detection of lesions in colonoscopy is subject to uncertainties due to inaccurate measurements and lack of precise modelling of lesion image characteristics (this is especially true for small size lesions) [19]. Given a colonoscopy image, the "true" features associated with the physical surface properties of the tissue are not exactly known to the system developer. Usually, one or more feature–extraction models [4,16,20] are used to provide values for each feature's parameters. The findings are then used to infer the correct interpretation.

In this work, we combine texture segmentation with neural networks for the automatic detection of lesions in colonoscopy images and video sequences.

The following subsections describe the various computational methods and principles that we have considered in developing our approach.

3.1 Texture Classification

Texture plays an important role in the characterization of regions in digital images. It carries information about the microstructure of the regions and the distribution of the grey levels. Texture is an inherent property of any image and of medical images in particular, given that the tissue itself carries a dominant textural appearance.

The classification of image regions within colonoscopy images can be treated as a texture classification problem by exploiting the textural characteristics of the corresponding regions for the discrimination between lesions and normal tissue samples. Automated classification and identification of colonic carcinoma using microscopy images have been proposed by [21], but the use of clinical endoscopy video frames for the identification of colonic tumors has been considered only in limited instances [4,17,18,22,23]. Along this line of research, this paper makes use of texture information for the detection of malignant regions in colonoscopy images by employing some quantitative description of a texture.

Among a large variety of texture models, e.g. structural [24], statistical [25–27] and random process [28], this work uses statistical measurements based on second order statistics [27]. These statistical descriptors have been estimated using the method of cooccurrence matrices applied to each region of an image. This method evaluates a series of matrices that describe the spatial variation of grey level values within a local area.

In our experiments we have used the image data management facilities of CoLD [4] to compute four cooccurrence matrices for each sample area with a displacement of one pixel and angles of 0, 45, 90, 135 degrees. In this way, four features have been computed on each matrix to produce a 16-dimensional feature vector describing each tissue sample, namely the angular second moment, correlation, inverse difference moment, and entropy, as defined by Haralick [27] (see [4] for details).

Let us consider an MLP whose l-th layer contains N_l neurons, $l = 1, \dots, M$. Batch learning is realized by minimizing the error function E defined by:

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{N_M} \left(y_{j,p}^{M} - t_{j,p} \right)^2 , \tag{1}$$

where $\left(y_{j,p}^{M} - t_{j,p} \right)^2$ is the squared difference between the actual output value at the j-th output layer neuron for the pattern p and the target output value, and p is an index over input–output pairs. The function E also provides the error surface over the weight space.

The minimization of function E corresponds to updating the weights by epoch, and requires a sequence of weight vectors $\{w^k\}_{k=0}^{\infty}$, where k indicates epochs. Successful training implies that $\{w^k\}_{k=0}^{\infty}$ converges to the point $w^\star$ that minimizes E. The gradient–based Back–Propagation (BP) training algorithm minimizes the error function using the steepest descent method with constant, heuristically chosen, learning rate.

Several BP algorithms with adaptive learning rate have been proposed in order to accelerate the training phase. In this work we consider the standard batch BP and four other methods of this class and investigate their performance in a distributed architecture.

The first BP variant that will be considered was proposed in [29,30]. It is a simple, heuristic, strategy for accelerating the BP algorithm that is based on the use of a momentum term (BPM). The second method was suggested by Vogl [31]. It increases the convergence of the BP by adapting the learning rate at each epoch, in such a way that monotone decrease of the error is enforced (VMRZA). To this end, Vogl et al. proposed to start training with a small learning rate and increase it by multiplying it with 1.05, if successive epochs reduce the error, or rapidly decrease it by multiplying it with 0.7, if a significant error increase occurs.

Another approach is based on the use of nonmonotone strategies for adapting the learning rate, i.e. deterministic adaptive training algorithms in which error function values are allowed to increase at some epochs [32]. This approach exploits the accumulated information with regard to previous error function values and provides the ability to handle large learning rates. Additionally, it alleviates problems generated by poor selection of the user–defined learning parameters, such as decreased rate of convergence, or even divergence and premature saturation [33]. Along this line, we investigate the use of the recently proposed NonMonotone Back–Propagation with Variable Stepsize (NMBPVS), [32], which exploits the local shape of the error surface by estimating the Lipschitz constant at each epoch, and setting the learning rate accordingly. Lastly, the NonMonotone Barzilai and Borwein BP algorithm (NMBBP), which uses an adaptive learning rate that is calculated by a two–point approximation to the secant equation [34,35], is applied.

4 A Distributed Architecture

Parallel processing, i.e. the method of having many small tasks with the aim of solving one large problem, has emerged as a key enabling technology in modern computing [36]. The past several years have witnessed an ever–increasing acceptance and adoption of parallel processing both for high–performance scientific computing, and for more "general–purpose" applications, as a result of the demand for higher performance, lower cost, and sustained productivity. This worldwide acceptance has been facilitated by two major developments: (a) Massively Parallel Processors (MPPs), and (b) the widespread use of Distributed Computing.

This section gives a brief introduction to distributed computing and describes in detail the methodology of our distributed architecture.

4.1 Distributed Computing and the Parallel Virtual Machine

Distributed Computing is a process whereby computers connected by a network are used collectively to solve a single problem [36]. The combined computational resources of several general–purpose workstations, interconnected with a high–speed local area network, may exceed the power of a single high performance computer.

The most critical factor in parallel processing is the high cost of the hardware. Large MPPs typically cost more than $10 million. In contrast with MPPs, distributed computing allows users running their problems on a local set of existing computers with a very little cost. Even building a PVM using dedicated computers has a reduced cost. The cost of a 15–node system is less than $10,000 due to the use of Beowulf–style nodes [37,38]. It must be noted that when using Beowulf nodes only the master node needs hard disk, video display, monitor and keyboard. The cost of materials for the PVM topology used in our experiments is exhibited in Section 4.2 below.

The Parallel Virtual Machine is a de facto standard message passing interface. It is an integrated set of software tools and libraries that emulates a general–purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architectures [9]. PVM is designed to link computing resources and provide users with a parallel platform for running their computer applications, irrespective of the number of different computer architectures and their locations. Notice that once built, the PVM can be used for any CPU intensive computational task [10,11]. PVM is capable of harnessing the combined resources of typically heterogeneous networked computing platforms to deliver high levels of performance and functionality.

The PVM system uses the message–passing model to allow programmers to exploit the distributed computing across a wide variety of computer architectures, including MPPs. PVM's key concept is that it makes a collection of computers to appear as one large *virtual* machine, hence its name [9].

4.2 PVM–based Training Methodology

The general use of an MLP consists of a training phase followed by a classification phase. The training phase, usually, involves an unconstrained optimization procedure of the BP class, and consists of the following steps:

(1) Presentation to the MLP of all the training sets (patterns) and computation of the activations of the network.
(2) Computation of the error function based on the activations (usually the sum of squared differences between the actual and the desired output).
(3) Computation or approximation of the gradients of the error function at a point in the weight space.
(4) Adaptation of the weights of the MLP according to the training algorithm used.

Training can be very time consuming, as a feasible minimizer in the high–dimensional weight space is sought, and, in general, the error function possesses complicated surface with multitudes of local minima and broad flat regions adjoined to narrow steep ones. On the other hand, the classification of an unknown test vector is extremely fast, since it requires only the propagation of the test vector through the neural network.

In the algorithm model shown above, Steps 1, 2 and 3 can be easily performed in parallel, if the training set is partitioned across multiple processors. On the other hand, Step 4 is better performed using only one processor, after the partially evaluated error function and gradient values are sent to it and accumulated.

Although MLPs have been widely used in many application areas, real world problems demand an increasing amount of computational resources. However, not many neural network researchers have access to a high performance parallel machine [39]. On the other hand, most of the researchers have access to networked workstations which can be easily used in a distributed computing architecture [40].

Below, we construct a parallel procedure that uses the well known master–slave computational model [9]. Our implementation is based on partitioning the training set across multiple processors on the slave nodes. This results in the distributed evaluation of the error function and gradient of the MLP. The following subsections describe the algorithms for the master and slave nodes and provide details for our implementation.

4.2.1 The Algorithm of the Master Node

At the beginning of the procedure, the master node adds the slave nodes to the PVM, spawns the slave tasks, partitions the initial training set into subsets (one for each slave node), and sends the network architecture and the partitions of the training set to the slaves. Then, receives from each slave node the corresponding portion of the error function and gradient, and accumulates them to find the complete error function and gradient values.

Note that the error function values are sent to the master only if the employed training algorithm uses them; otherwise only the partial gradient values are communicated. Finally, the weights are updated (any batch training algorithm can be used for this task), and the new weights are sent to the slave nodes for the next epoch. When the termination condition is fulfilled, the master node sends termination signals to the slaves and shuts the PVM down. Below, a high level description of the algorithm that runs on the master is presented.

```
Procedure master
\* Spawn slave procedures *\
InitializeAllSlaves;
\* Load patterns, initialize weights
   and MLP architecture *\
InitializeMLP;
\* Divide up the training set among the slaves *\
PartitionTrainingSet;
For slave := 1 to numOfSlaves do
   \* Send MLP architecture to slave *\
   SendMLP;
   \* Send training subset to slave *\
   SendPatterns;
```

```
Repeat
   For slave := 1 to numOfSlaves do
      \* Receive error function values from slaves *\
      RcvPartialErrorFunction;
      \* Receive gradient values from slaves *\
      RcvPartialGradient;
   \* Calculate the error function value over
      the entire training set *\
   AccumulateErrorFunction;
   \* Calculate the gradient vector *\
   AccumulateGradient;
   \* Calculate the new weight vector *\
   AdaptWeights;
   For slave := 1 to numOfSlaves do
      \* Send the new weight vector to slaves *\
      SendWeights;
\* Check the termination condition *\
Unitl TerminationCondition;
\* Kill the slave processes *\
ShutDownSlaves;
\* Kill the PVM *\
ShutDownPVM;
```

The process of receiving the partial error function and gradient values can be realized in a synchronous or an asynchronous mode. When the synchronous mode is selected the master is forced to communicate with the slaves in a specific order. On the other hand, in asynchronous mode the communications are performed in a first–come, first–serve basis. Obviously, the asynchronous mode is the preferred one, since the master does not have to wait for a slower slave, but instead can continue gathering information from the other slaves.

4.2.2 The Algorithm of the Slave Nodes

Each slave node initially receives the MLP architecture and a subset of the training set. Then, it calculates the partial error function and gradient values by means of a forward and a backward pass, and sends these values to the master node. Finally, the master node sends the updated weights for the next epoch. Below, we provide a high level description of the algorithm running on the slave.

```
procedure slave
\* Receive MLP architecture from master *\
RcvMLP;
```

```
\* Receive training subset from master *\
RcvPatterns;
Repeat
   \* Calculate partial error function values *\
   CalculatePartialErrorFunction;
   \* Calculate partial gradient values *\
   CalculatePartialGradient;
   \* Send error function values to master *\
   SendPartialErrorFunction;
   \* Send gradient values to master *\
   SendPartialGradient;
   \* Receive new weight vector from master *\
   RcvWeights;
Until ShutDown;
```

4.3 Implementation Details

In this work, the combination of texture segmentation and neural networks is employed for the automatic detection of lesions in colonoscopy images and video sequences. The overall procedure is illustrated in Figure 1.

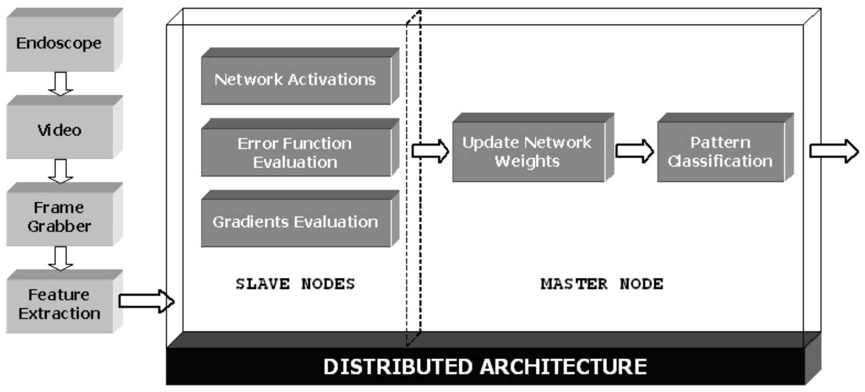

Fig. 1. Illustration of proposed methodology for image–guided diagnosis.

Below, we provide technical details and cost estimates for the PVM implementation. The setup of the PVM is relatively easy with the use of existing workstations. In the PVM system used in our experiments, 15 slaves and one master node were connected using a 100 Mbps Ethernet switch. A daemon process running in the background of each node forms the Parallel Virtual Machine. The daemons are responsible for spawning the tasks (program execution) on the host machines, the synchronization, and the communication between the tasks.

The cost of materials for the 15-node system is shown in Table 1. The total price for the entire system is less than $10,000 (prices as of early 2004). It must be noted that only the master node needs hard disk, video display, monitor and keyboard. This is possible because of the use of *Beowulf–style* nodes. Additional information about the Beowulf Project and further details for parallel computer systems, can be found on the WWW at: `http://www.beowulf.org`.

Table 1
Cost of materials for a 15-node Parallel Virtual Machine (prices as of 2004).

Quantity	*Item*	*Unit price*	*Total*
15	Intel Pentium 4 2.8Ghz processor and appropriate motherboard	$500	$7,500
15	512 MB of SDRAM	$40	$600
15	10/100 Mbps Ethernet network interface card	$10	$150
15	Tower case with 300 Watt power supply and fans	$45	$675
1	10/100 Mbps Ethernet 24–port Switch	$330	$330
16	Ethernet cables	$5	$80
1	Master computer with Intel Pentium 4 3.06Ghz, 512 MB of RAM, 10/100 Mbps Ethernet network card, 60 GB hard disk, video display, mouse, monitor, keyboard	$650	$650
16	Linux Operating System	$0	$0
		TOTAL	$9,985

The operating system used in our implementation was Linux, which is the most common operating system for individual nodes of Beowulf–style parallel computer systems. Another reason for choosing Linux is that it is open–source (its source code is provided) and free; no licence is required.

5 Experimental Results

The proposed distributed computing methodology was applied for the detection of malignant regions in colonoscopic video sequences. The aim of the experiment was to perform a low level test of the system and explore the applicability of our methodology in a real life diagnostic task.

Textures from normal and abnormal tissue samples were randomly chosen from four frames of the same video sequence, which exhibited resolution change, different perceptual direction of the physician, different diffused light conditions, and were used for training the MLP to discriminate between malignant and normal regions using the distributed architecture. No pre–filtering of the images, or post–processing of the results were applied as the aim was only to test the PVM–methodology and not to optimize the classifier.

The training set was generated by applying the cooccurrence matrices method described in previous section. More specifically, the endoscopic images were separated into windows of size 16×16 pixels with 8 pixels overlap. Then the cooccurrence matrices algorithm was used to gather information regarding each pixel in an image window [4,17]. The 16-dimensional feature vectors created for each window was used as the input of an MLP with 16 inputs, 30 hidden nodes, and 2 outputs (540 weights and 32 biases); this MLP architecture had been found to perform very well in preliminary experiments. The MLP was trained to discriminate between normal and abnormal image regions using 1200 randomly selected patterns from four video frames. The training procedure stopped when the MLP exhibited 3% misclassifications on the entire training set.

Using the PVM–based training methodology, the following algorithms were implemented:

- the Back–Propagation (BP),
- the momentum BP (MBP) [29],
- the adaptive BP (VMRZA) [31],
- the Non–Monotone BP with Variable Stepsize (NMBPVS) [41],
- the Non–Monotone Barzilai–Borwein back–Propagation NMBBP [41].

The algorithms were tested using the same initial weights, initialized by the Nguyen–Widrow method [42], and received the same sequence of input patterns. The weights of the MLP were updated only after the entire set of patterns to be learned was presented and processed in parallel.

Table 2 summarizes the performance of the algorithms for simulations that reached solution. The reported parameters are: *min* the minimum number of epochs, *mean* the mean value of epochs, *max* the maximum number of epochs, *s.d.* the standard deviation, and *succ.* the simulations succeeded out of 100 trials.

As is the case with all practical neural network training, the aim is to train the MLPs to achieve a balance between the ability to respond correctly to the input data used for the training (memorization) and the ability to give correct responses to input that is similar, but not identical, to that used in training (generalization). To this end, to test the generalization performance of the trained MLPs, approximately 16,000 test patterns were created. This test set constitutes the whole image region in each of the four frames and contains normal and abnormal samples. In Table 3, the average generalization capability of the algorithms on the test set is exhibited.

Table 2
Results of the proposed distributed implementation of the training algorithms.

Algorithm	min	mean	max	s.d.	succ.
BP	7697	8505	9314	1143	20%
BPM	5685	8500	9315	952	32%
VMRZA	453	734	1055	249	99%
NMBBP	261	374	515	129	100%
NMBPVS	263	656	955	227	100%

Table 3
Generalization of the distributed implementation of the training algorithms.

Algorithm	Generalization (%)
BP	78.1%
BPM	78.1%
VMRZA	79.1%
NMBBP	83.9%
NMBPVS	85.1%

Finally, we have tried to determine the average speedup achieved by employing the proposed distributed implementation, relative to a single processor utilization. Several factors can influence the speedup, such as the local area network load and the CPU load due to system or other users' tasks. Nevertheless, the speedup results indicate that when using more than three slave nodes the combined processing power of the PVM overbalances the overhead due to its initialization and process communication, and a speedup is always possible. In Figure 2 the speedup versus the number of processors is plotted.

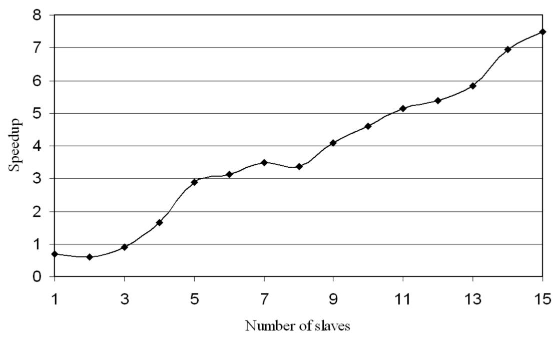

Fig. 2. The speedup achieved by the proposed distributed implementation versus the number of processors used in the simulation.

Thus, the speedup is considerable and worth the minimal effort of developing the PVM implementation of the learning algorithm, although it is not analogous to the number of slaves used. Obviously, this was expected due to the overhead introduced by the local area network and the PVM itself.

6 Conclusions and future plans

Research in computing, imaging and miniaturization has made minimally invasive surgery practical and has opened up new areas of research for diagnosis and treatment. An increasing need for large computing resources is appearing in hospitals for image–guided diagnosis and surgery, simulation of medical treatments and surgeries, advanced medical imaging applications, and accessibility of large amounts of data in heterogeneous formats from distributed sources. Distributed computing can facilitate the deployment of these advanced medical applications.

In this paper, the combined computational resources of several general purpose workstations, interconnected with a local area network have been exploited to implement neural network learning algorithms in a distributed architecture for image–guided diagnosis of lesions in colonoscopy video sequences. Each workstation of the proposed distributed architecture handles intensive computational tasks efficiently, without the use of frequent process synchronization. Furthermore, the PVM performance in the experiments was stable and predictable.

Simulation results have shown that speedups are always possible and justify the extra effort of parallelizing the learning algorithm, especially when large MLP architectures and large training sets are used. Our experience is that the speedup achieved does not affect the generalization performance of the neural networks with respect to the single classifier implementation.

Acknowledgments

The authors gratefully acknowledge the contribution of Dr S. Karkanis (Department of Informatics and Computer Technology, Technological Educational Institute of Lamia, Greece) and Dr D. Iakovidis (RTD–Image Group, Department of Informatics and Telecommunications, University of Athens, Greece) in the acquisition of the data and Mr. N. Nousis for his valuable comments and helpful suggestions on earlier drafts of this paper.

The authors would like to thank the European Social Fund, Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the Program PYTHAGORAS for funding the above work. Dr V.P. Plagianakos and Prof. M.N. Vrahatis acknowledge the financial support of the University of Patras Research Committee through a "Karatheodoris" research grant.

References

[1] C. Giess, A. Mayer, H. Evers, and H.P. Meinzer, Medical Image Processing and Visualization on Heterogenous Clusters of Symmetric Multiprocessors using MPI and POSIX Threads, in Proc. of IPPS/SPDP 98, pp. 233–237, (Orlando, 1998).

[2] V. Breton, R. Medina, and J. Montagnat, DataGrid, Prototype of a Biomedical Grid, Methods of Information in Medicine, 42 (2003) 1–5.

[3] K. Cleary, M. Clifford, M. Freedman, J. Zeng, S.K. Mun, V. Watson, and F. Henderson, Technology improvements for image–guided and minimally invasive spine procedures, IEEE Transactions on Information Technology in Biomedicine, 6 (2002) 249–261.

[4] D.E. Maroulis, D.K. Iakovidis, S.A. Karkanis, D.A. Karras, CoLD: A versatile detection system for colorectal lesions in endoscopy video-frames, Computer Methods and Programs in Biomedicine, 70 (2003) 99–186.

[5] S. Delp, P. Loan, C. Basdogan, and J.M. Rosen, Surgical simulation: An emerging technology for training in emergency medicine, Presence, 6 (1997) 147–159.

[6] A.M. Eldeib, M.N. Ahmed, A.A. Farag and C.B. Sites, A Web based System for Surgical Planning and Simulation, Proc. of SPIE, 3517 (1998) 273–283.

[7] A.A. Farag and C.B. Sites, Virtual Endoscopy: Modeling the Navigation in 3D Brain Volumes,in Proc. of the International Conference on Biomedical Engineering, (Egypt, 2002).

[8] U. Kuhnapfel, H.K. Cakmak, and H. Maab, Endoscopic surgery training using virtual reality and deformable tissue simulation, Computers & Graphics, 24 (2000) 671–682.

[9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing, (MIT Press, Cambridge, 1994).

[10] V.P. Plagianakos and M.N. Vrahatis, Parallel Evolutionary Training Algorithms for "Hardware–Friendly" Neural Networks, Natural Computing, 1 (2002) 307–322.

[11] V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis, Evolutionary training of hardware realizable multilayer perceptrons, Neural Computing and Applications, (2006) in press.

[12] P.M. Delaney, G.D. Papworth, and R.G. King, Fibre optic confocal imaging (FOCI) for in vivo subsurface microscopy of the colon, eds. V.R. Preedy and R.R. Watson, Methods in disease: Investigating the gastrointestinal tract, (Greenwich Medical Media, London, UK, 1998).

[13] American Cancer Society, Cancer facts and figures, American Cancer Society, Atlanta, Georgia, publication no.5008.00, (2000).

[14] S. Parker, T. Tong, S. Bolden, and P. Wingo, Cancer Statistics, CA Cancer Journal for Clinicians, 47 (1997) 5–27.

[15] D. Rex, R. Weddle, D. Pound, K. O'Connor, R. Hawes, R. Dittus, J. Lappas, and L. Lumeng, Flexible sigmoidoscopy plus air contrast barium enema versus colonoscopy for suspected lower gastrointestinal bleeding, Gastroenterology, 98 (1990) 855–861.

[16] S.A. Karkanis, D.K. Iakovidis, D.A. Karras and D.E. Maroulis, Computer Aided Tumor Detection in Endoscopic Video using Color Wavelet Features, IEEE Transactions in Information Technology in Biomedicine, 7 (2003) 141–152.

[17] S.A. Karkanis, G.D. Magoulas, and N. Theofanous, Image recognition and neuronal networks: Intelligent systems for the improvement of imaging information, Minimally Invasive Therapy & Allied Technologies, 9 (2000) 225–230.

[18] G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis, Neural Network-based Colonoscopic Diagnosis Using On-line Learning and Differential Evolution, Applied Soft Computing, 4 (2004) 369–379.

[19] C.K. Kwoh, Probabilistic reasoning from correlated objective data, Ph.D. Thesis, (Imperial College, London, UK, 1995).

[20] C.G. Looney, Pattern recognition using neural networks, (Oxford University Press, Oxford, UK, 1997).

[21] A.N. Esgiar, R.N.G. Naguib, B.S. Sharif, M.K. Bennett, and A. Murray, Microscopic image analysis for quantitative measurement and feature identification of normal and cancerous colonic mucosa, IEEE Trans. Inform. Technol. Biomed., 2 (1998) 197–203.

[22] S.A. Karkanis, G.D. Magoulas, D.K. Iakovidis, D.A. Karras, and D.E. Maroulis, Evaluation of textural feature extraction schemes for neural network–based interpretation of regions in medical images, in Proc. of the IEEE International Conference on Image Processing (ICIP), pp. 281–284, (Thessaloniki, Greece, 2001).

[23] S.A. Karkanis, G.D. Magoulas, D.K. Iakovidis, D.E. Maroulis, and N. Theofanous, Tumor recognition in endoscopic video images, in Proc. of the 26th EUROMICRO Conference, pp. 423–429, (Maastricht, Netherlands, 2000).

[24] J.G. Daugman, Complete discrete 2D Gabor transforms by neural networks for image analysis and compression, IEEE Trans. Acoustic, Speech and Signal Processing, 36 (1988) 1169–1179.

[25] C.H. Chen, A study of texture classification using spectral features, in Proc. of the International Conference on Pattern Recognition, pp. 1074–1077, (Munich, Germany, 1982).

[26] C.C. Gotlieb and K. Kreyszig, Texture descriptors based on cooccurrence matrices, Computer Vision, Graphics and Image Processing, 51 (1990) 70–86.

[27] R.M. Haralick, Statistical and structural approaches to texture, IEEE Proc., 67 (1979) 786–804.

[28] G.R. Cross and A.K. Jain, Markov random field texture models, IEEE Transactions on Pattern Analysis and Machine Intelligence, 5 (1983) 25–39.

[29] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks, 1 (1988) 295–307.

[30] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation, eds. D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, pp. 318–362, (MIT Press, Cambridge, Massachusetts, 1986).

[31] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink, and D.L. Alkon, Accelerating the convergence of the back–propagation method, Biological Cybernetics, 59 (1988) 257–263.

[32] V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis, Deterministic Nonmonotone Strategies for Effective Training of Multi–Layer Perceptrons, IEEE Transactions on Neural Networks, 13 (2002) 1268–1284.

[33] Y. Lee, S.-H. Oh and M.W. Kim, An analysis of premature saturation in backpropagation learning, Neural Networks, 6 (1993) 719–728.

[34] J. Barzilai and J.M. Borwein, Two point step size gradient methods, IMA Journal of Numerical Analysis, 8 (1998) 141–148.

[35] V.P. Plagianakos, D.G. Sotiropoulos, and M.N. Vrahatis, Automatic adaptation of learning rate for backpropagation neural networks, ed. N.E. Mastorakis, Recent Advances in Circuits and Systems, pp. 337–341 (World Scientific, Singapore, 1998).

[36] C. Leopold, Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches, (John Wiley & Sons, 2000).

[37] The Beowulf Project, http://www.beowulf.org, last accessed 15/06/2004.

[38] T.L. Sterling, J. Salmon, D.J. Becker, and D.F. Savarese, How to build a Beowulf: A Guide to Implementation and Application of PC Clusters, (MIT Press, Cambridge, 1999).

[39] L. Coetzee and E.C. Botha, An analysis of coarse–grain parallel training of a neural net, Network: Computation in Neural Systems, 6 (1995) 73–91.

[40] D. Anguita, A. Boni, and G. Parodi, A case study of distributed high–performance computing system for neurocomputing, Journal of Systems Architecture, 46 (2000) 429–438.

[41] V.P. Plagianakos, M.N. Vrahatis, and G.D. Magoulas, Nonmonotone Methods for Backpropagation Training with Adaptive Learning Rate, in Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN'99), pp. 2219–2223, (Washington D.C., 1999).

[42] D. Nguyen and B. Widrow, Improving the learning speed of 2–layer neural network by choosing initial values of the adaptive weights, in Proc. of the IEEE First International Joint Conference on Neural Networks, pp. 21–26, (1990).