Chapter 7

Parallel Implementation of Morphological Neural Networks for Hyperspectral Image Analysis

Javier Plaza,
University of Extremadura, Spain

Rosa Pérez,
University of Extremadura, Spain

Antonio Plaza,
University of Extremadura, Spain

Pablo Martínez,
University of Extremadura, Spain

David Valencia,
University of Extremadura, Spain

Contents

Improvement of spatial and spectral resolution in latest-generation Earth observation instruments is introducing extremely high computational requirements in many remote sensing applications. While thematic classification applications have greatly benefited from this increasing amount of information, new computational requirements have been introduced, in particular, for hyperspectral image data sets with

hundreds of spectral channels and very fine spatial resolution. Low-cost parallel computing architectures such as heterogeneous networks of computers have quickly become a standard tool of choice for dealing with the massive amount of image data sets. In this chapter, a new parallel classification algorithm for hyperspectral imagery based on morphological neural networks is presented and discussed. The parallel algorithm is mapped onto heterogeneous and homogeneous parallel platforms using a hybrid partitioning scheme. In order to test the accuracy and parallel performance of the proposed approach, we have used two networks of workstations distributed among different locations, and also a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center in Maryland. Experimental results are provided in the context of a real agriculture and farming application, using hyperspectral data acquired by the Airborne Visible Infra-Red Imaging Spectrometer (AVIRS), operated by the NASA Jet Propulstion Laboratory, over the valley of Salinas in California.

7.1 Introduction

Many international agencies and research organizations are currently devoted to the analysis and interpretation of high-dimensional image data collected over the surface of the Earth [1]. For instance, NASA is continuously gathering hyperspectral images using the Jet Propulsion Laboratory's Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) [2], which measures reflected radiation in the wavelength range from 0.4 to 2.5 μm using 224 spectral channels at a spectral resolution of 10 nm. The incorporation of hyperspectral instruments aboard satellite platforms is now producing a near-continual stream of high-dimensional remotely sensed data, and cost-effective techniques for information extraction and mining from massively large hyperspectral data repositories are highly required [3]. In particular, although it is estimated that several Terabytes of hyperspectral data are collected every day, about 70% of the collected data is never processed, mainly due to the extremely high computational requirements.

Several challenges still remain open in the development of efficient data processing techniques for hyperspectral image analysis [1]. For instance, previous research has demonstrated that the high-dimensional data space spanned by hyperspectral data sets is usually empty [4], indicating that the data structure involved exists primarily in a subspace. A commonly used approach to reduce the dimensionality of the data is the principal component transform (PCT) [5]. However, this approach is characterized by its global nature and cannot preserve subtle spectral differences required to obtain a good discrimination of classes [6]. Further, this approach relies on spectral properties of the data alone, thus neglecting the information related to the spatial arrangement of the pixels in the scene. As a result, there is a need for feature extraction techniques able to integrate the spatial and spectral information available from the data simultaneously [5].

While such integrated spatial/spectral developments hold great promise in the field of remote sensing data analysis, they introduce new processing challenges [7, 8]. The concept of Beowulf cluster was developed, in part, to address such challenges [9, 10]. The goal was to create parallel computing systems from commodity components to satisfy specific requirements for the earth and space sciences community. Although most dedicated parallel machines employed by NASA and other institutions during the last decade have been chiefly homogeneous in nature, a current trend is to utilize heterogeneous and distributed parallel computing platforms [11]. In particular, computing on heterogeneous networks of computers (HNOCs) is an economical alternative that can benefit from local (user) computing resources while, at the same time, achieving high communication speed at lower prices. The properties above have led HNOCs to become a standard tool for high-performance computing in many ongoing and planned remote sensing missions [3, 11].

To address the need for cost-effective and innovative algorithms in this emerging new area, this chapter develops a new parallel algorithm for the classification of hyperspectral imagery. The algorithm is inspired by previous work on morphological neural networks, such as autoassociative morphological memories and morphological perceptrons [12], although it is based on different concepts. Most importantly, it can be tuned for very efficient execution on both HNOCs and massively parallel, Beowulf-type commodity clusters. The remainder of the chapter is structured as follows.

- Section 7.2 describes the proposed heterogeneous parallel algorithm, which consists of two main processing steps: 1) a parallel morphological feature extraction taking into account the spatial and spectral information, and 2) robust classification using a parallel multi-layer neural network with back-propagation learning.
- Section 7.3 describes the algorithm's accuracy and parallel performance. Classification accuracy is discussed in the context of a real application that makes use of hyperspectral data collected by the AVIRIS sensor, operated by NASA's Jet Propulsion Laboratory, to assess agricultural fields in the valley of Salinas, California. Parallel performance in the context of the above-mentioned application is then assessed by comparing the efficiency achieved by an heterogeneous parallel version of the proposed algorithm, executed on a fully heterogeneous network, with the efficiency achieved by its equivalent homogeneous version, executed on a fully homogeneous network with the same aggregate performance as the heterogeneous one. For comparative purposes, performance data on Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center, are also given.
- Finally, Section 7.4 concludes with some remarks and hints at plausible future research, including implementations of the proposed parallel algorithm on specialized hardware architectures.

7.2 Parallel Morphological Neural Network Algorithm

This section describes a new parallel algorithm for the analysis of remotely sensed hyperspectral images. Before describing the two main steps of the algorithm, we first formulate a general optimization problem in the context of HNOCs, composed of different-speed processors that communicate through links at different capacities [11]. This type of platform can be modeled as a complete graph, $G = (P, E)$, where each node models a computing resource p_i weighted by its relative cycle-time w_i. Each edge in the graph models a communication link weighted by its relative capacity, where c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j. We also assume that the system has symmetric costs, i.e., $c_{ij} = c_{ji}$. Under the above assumptions, processor p_i will accomplish a share of $\alpha_i \times W$ of the total workload W, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. With the above assumptions in mind, an abstract view of our problem can be simply stated in the form of a client-server architecture, in which the server is responsible for the efficient distribution of work among the P nodes, and the clients operate with the spatial and spectral information contained in a local partition. The partitions are then updated locally and the resulting calculations may also be exchanged between the clients, or between the server and the clients. Below, we describe the two steps of our parallel algorithm.

7.2.1 Parallel Morphological Algorithm

The proposed feature extraction method is based on mathematical morphology [13] concepts. The goal is to impose an ordering relation (in terms of spectral purity) in the set of pixel vectors lying within a spatial search window (called a structuring element) designed by B [5]. This is done by defining a cumulative distance between a pixel vector $f(x, y)$ and all the pixel vectors in the spatial neighborhood given by B (B-neighborhood) as follows: $D_B[f(x, y)] = \sum_i \sum_j \text{SAD}[f(x, y), f(i, j)]$, where (x, y) refers to the spatial coordinates in the B-neighborhood and SAD is the spectral angle distance [1]. From the above definitions, two standard morphological operations called erosion and dilation can be respectively defined as follows:

$$(f \otimes B)(x, y) = argmin_{(s,t)\in Z^2(B)} \sum_s \sum_t \text{SAD}(f(x, y), \quad f(x + s, y + t)) \tag{7.1}$$

$$(f \oplus B)(x, y) = argmax_{(s,t)\in Z^2(B)} \sum_s \sum_t \text{SAD}(f(x, y), \quad f(x - s, y - t)) \tag{7.2}$$

Using the above operations, the opening filter is defined as $(f \circ B)(x, y) = [(f \otimes C) \oplus B](x, y)$ (erosion followed by dilation), while the closing filter is defined as $(f \bullet B)(x, y) = [(f \oplus C) \otimes B](x, y)$ (dilation followed by erosion). The composition of the opening and closing operations is called a spatial/spectral profile,

which is defined as a vector that stores the relative spectral variation for every step of an increasing series. Let us denote by $\{(f \circ B)^{\lambda}(x, y)\}, \lambda = \{0, 1, ..., k\}$, the *opening series* at $f(x, y)$, meaning that several consecutive opening filters are applied using the same window B. Similarly, let us denote by $\{(f \bullet B)^{\lambda}(x, y)\}, \lambda = \{0, 1, ..., k\}$, the *closing series* at $f(x, y)$. Then, the spatial/spectral profile at $f(x, y)$ is given by the following vector:

$$p(x, y) = \{\text{SAD}((f \circ B)^{\lambda}(x, y), (f \circ B)^{\lambda-1}(x, y))\} \cup \{\text{SAD}((f \bullet B)^{\lambda}(x, y), (f \bullet B)^{\lambda-1}(x, y))\} \quad (7.3)$$

Here, the step of the opening/closing series iteration at which the spatial/spectral profile provides a maximum value gives an intuitive idea of both the spectral and spatial distributions in the B-neighborhood [5]. As a result, the profile can be used as a feature vector on which the classification is performed using a spatial/spectral criterion.

In order to implement the algorithm above in parallel, two types of partitioning can be exploited:

- Spectral-domain partitioning subdivides the volume into small cells or subvolumes made up of contiguous spectral bands, and assigns one or more subvolumes to each processor. With this model, each pixel vector is split amongst several processors, which breaks the spectral identity of the data because the calculations for each pixel vector (e.g., for the SAD calculation) need to originate from several different processing units.
- Spatial-domain partitioning provides data chunks in which the same pixel vector is never partitioned among several processors. With this model, each pixel vector is always retained in the same processor and is never split.

In this work, we adopt a spatial-domain partitioning approach for several reasons:

- A first major reason is that the application of spatial-domain partitioning is a natural approach for morphological image processing, as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure, as indicated in the previous subsection.
- A second reason has to do with the cost of inter-processor communication. In spectral-domain partitioning, the window-based calculations made for each hyperspectral pixel need to originate from several processing elements, in particular, when such elements are located at the border of the local data partitions (see Figure 7.1), thus requiring intensive inter-processor communication.

However, if redundant information such as an overlap border is added to each of the adjacent partitions to avoid access from outside the image domain, then boundary data to be communicated between neighboring processors can be greatly minimized. Such an overlapping scatter would obviously introduce redundant computations, since the intersection between partitions would be non-empty. Our implementation makes

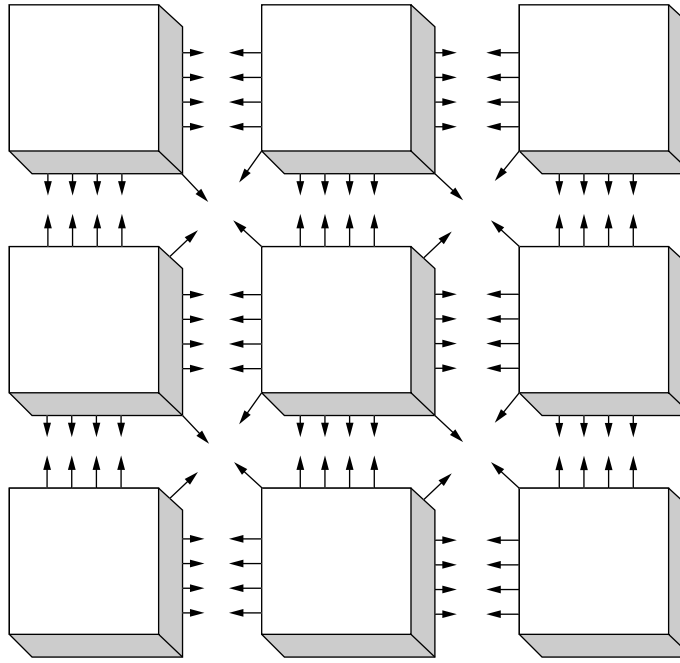

Figure 7.1 Communication framework for the morphological feature extraction algorithm.

use of a constant structuring element B (with size of 3×3 pixels) that is repeatedly iterated to increase the spatial context, and the total amount of redundant information is minimized. To do so, we have implemented a special 'overlapping scatter' operation that also sends out the overlap border data as part of the scatter operation itself (i.e., redundant computations replace communications).

To implement the algorithm, we made use of MPI *derived datatypes* to directly scatter hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step. A comparison between the associative costs of redundant computations in overlap with the overlapping scatter approach, versus the communications costs of accessing neighboring cell elements outside of the image domain, has been presented and discussed in previous work [7].

A pseudo-code of the proposed HeteroMORPH parallel algorithm, specifically tuned for HNOCs, is given below:

Inputs: N-dimensional cube f, structuring element B.

Output: Set of morphological profiles for each pixel.

1. Obtain information about the heterogeneous system, including the number of processors, P; each processor's identification number, $\{p_i\}_{i=1}^{P}$; and processor cycle-times, $\{w_i\}_{i=1}^{P}$.

2. Using B and the information obtained in step 1, determine the total volume of information, R, that needs to be replicated from the original data volume, V, according to the data communication strategies outlined above, and let the total workload W to be handled by the algorithm be given by $W = V + R$.
3. Set $\alpha_i = \lfloor \frac{(P/w_i)}{\sum_{i=1}^{P}(1/w_i)} \rfloor$ for all $i \in \{1, ..., P\}$.
4. For $m = \sum_{i=1}^{P} \alpha_i$ to $(V + R)$, find $k \in \{1, .., P\}$ so that $w_k \cdot (\alpha_k + 1) = \min\{w_i \cdot (\alpha_i + 1)\}_{i=1}^{P}$ and set $\alpha_k = \alpha_k + 1$.
5. Use the resulting $\{\alpha_i\}_{i=1}^{P}$ to obtain a set of P spatial-domain heterogeneous partitions (with overlap borders) of W, and send each partition to processor p_i, along with B.
6. Calculate the morphological profiles $p(x, y)$ for the pixels in the local data partitions (in parallel) at each heterogeneous processor.
7. Collect all the individual results and merge them together to produce the final output.

A homogeneous version of the HeteroMORPH algorithm above can be simply obtained by replacing step 4 with $\alpha_i = P/w_i$ for all $i \in \{1, ..., P\}$, where w_i is the communication speed between processor pairs in the network, which is assumed to be homogeneous.

7.2.2 Parallel Neural Algorithm

In this section, we describe a supervised parallel classifier based on a multi-layer perceptron (MLP) neural network with back-propagation learning. This approach has been shown in previous work to be very robust for the classification of hyperspectral imagery [14]. However, the considered neural architecture and back-propagation-type learning algorithm introduce additional considerations for parallel implementations on HNOCs.

The architecture adopted for the proposed MLP-based neural network classifier is shown in Figure 7.2. As shown in the figure, the number of input neurons equals the number of spectral bands acquired by the sensor. In the case of PCT-based preprocessing or morphological feature extraction commonly adopted in hyperspectral analysis, the number of neurons at the input layer equals the dimensionality of feature vectors used for classification. The second layer is the hidden layer, where the number of nodes, M, is usually estimated empirically. Finally, the number of neurons at the output layer, C, equals the number of distinct classes to be identified in the input data. With the above architecture in mind, the standard back-propagation learning algorithm can be outlined by the following steps:

1. *Forward phase*. Let the individual components of an input pattern be denoted by $f_j(x, y)$, with $j = 1, 2, ..., N$. The output of the neurons at the hidden layer is obtained as: $H_i = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x, y))$ with $i = 1, 2, ..., M$, where $\varphi(\cdot)$ is the activation function and ω_{ij} is the weight associated to the connection between the i-th input node and the j-th hidden node. The outputs of the MLP

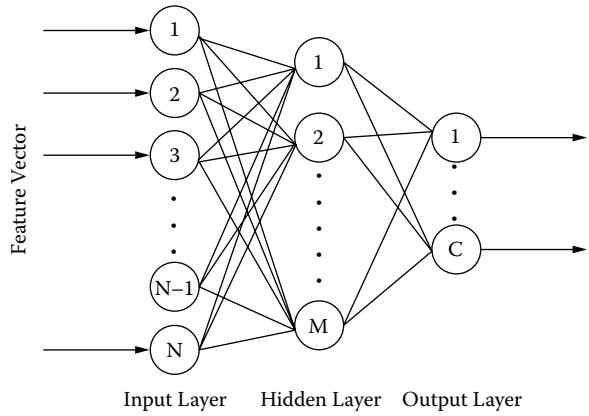

Figure 7.2 MLP neural network topology.

are obtained using $O_k = \varphi(\sum_{i=1}^{M} \omega_{ki} \cdot H_i)$, with $k = 1, 2, ..., C$. Here, ω_{ki} is the weight associated to the connection between the i-th hidden node and the k-th output node.

2. *Error back-propagation*. In this stage, the differences between the desired and obtained network outputs are calculated and back-propagated. The *delta* terms for every node in the output layer are calculated using $\delta_k^o = (O_k - d_k) \cdot \varphi'(\cdot)$, with $i = 1, 2, ..., C$. Here, $\varphi'(\cdot)$ is the first derivative of the activation function. Similarly, *delta* terms for the hidden nodes are obtained using $\delta_i^h = \sum_{k=1}^{C}(\omega_{ki} \cdot \delta_i^o) \cdot \varphi(\cdot))$, with $i = 1, 2, ..., M$.
3. *Weight update*. After the back-propagation step, all the weights of the network need to be updated according to the *delta* terms and to η, a learning rate parameter. This is done using $\omega_{ij} = \omega_{ij} + \eta \cdot \delta_i^h \cdot f_j(x, y)$ and $\omega_{ki} = \omega_{ki} + \eta \cdot \delta_k^o \cdot H_i$. Once this stage is accomplished, another training pattern is presented to the network and the procedure is repeated for all incoming training patterns.

Once the back-propagation learning algorithm is finalized, a classification stage follows, in which each input pixel vector is classified using the weights obtained by the network during the training stage [14].

Two different schemes can be adopted for the partitioning of the multi-layer perceptron classifier:

- The exemplar partitioning scheme, also called training example parallelism, explores data level parallelism and can be easily obtained by simply partitioning the training pattern data set. Each process determines the weight changes for a disjoint subset of the training population, and then changes are combined and applied to the neural network at the end of each epoch. This scheme requires a suitable large number of training patterns to take advantage of it, which is

not a very common situation in most remote sensing applications, as long as it is a very hard task to get ground-truth information for regions of interest in a hyperspectral scene.

- The hybrid partition scheme, on the other hand, relies on a combination of neuronal level as well as synaptic level parallelism [15], which allows one to reduce the processors' intercommunications at each iteration. In the case of neuronal parallelism (also called vertical partitioning), all the incoming weights to the neurons local to the processor are computed by a single processor. In synaptic level parallelism, each workstation will compute only the outgoing weight connections of the nodes (neurons) local to the processor. In the hybrid scheme, the hidden layer is partitioned using neuronal parallelism while weight connections adopt the synaptic scheme.

The parallel classifier presented in this section is based on a hybrid partitioning scheme, where the hidden layer is partitioned using neuronal level parallelism and weight connections are partitioned on the basis of synaptic level parallelism [16]. As a result, the input and output neurons are common to all processors, while the hidden layer is partitioned so that each heterogeneous processor receives a number of hidden neurons, which depends on its relative speed. Each processor stores the weight connections between the neurons local to the processor. Since the fully connected MLP network is partitioned into P partitions and then mapped onto P heterogeneous processors using the above framework, each processor is required to communicate with every other processor to simulate the complete network. For this purpose, each of the processors in the network executes the three phases of the back-propagation learning algorithm described above. The HeteroNEURAL algorithm can be summarized as follows:

Inputs: N-dimensional cube f, training patterns $f_j(x, y)$.

Output: Set of classification labels for each image pixel.

1. Use steps 1–4 of the HeteroMORPH algorithm to obtain a set of values $(\alpha_i)_{i=1}^{P}$, which will determine the share of the workload to be accomplished by each heterogeneous processor.
2. Use the resulting $(\alpha_i)_{i=1}^{P}$ to obtain a set of P heterogeneous partitions of the hidden layer and map the resulting partitions among the P heterogeneous processors (which also store the full input and output layers along with all connections involving local neurons).
3. *Parallel training.* For each considered training pattern, the following three parallel steps are executed:
   (a) *Parallel forward phase.* In this phase, the activation value of the hidden neurons local to the processors are calculated. For each input pattern, the activation value for the hidden neurons is calculated using $H_i^P = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x, y))$. Here, the activation values and weight connections of neurons present in other processors are required to calculate the activation values of output neurons according to $O_k^P = \varphi(\sum_{i=1}^{M/P} \omega_{ki}^P \cdot H_i^P)$,

with $k = 1, 2, ..., C$. In our implementation, broadcasting the weights and activation values is circumvented by calculating the partial sum of the activation values of the output neurons.

(b) *Parallel error back-propagation.* In this phase, each processor calculates the error terms for the local hidden neurons. To do so, *delta* terms for the output neurons are first calculated using $(\delta_k^o)^P = (O_k - d_k)^P \cdot \varphi'(\cdot)$, with $i = 1, 2, ..., C$. Then, error terms for the hidden layer are computed using $(\delta_i^h)^P = \sum_{k=1}^{P} (\omega_{ki}^P \cdot (\delta_k^o)^P) \cdot \varphi'(\cdot)$, with $i = 1, 2, ..., N$.

(c) *Parallel weight update.* In this phase, the weight connections between the input and hidden layers are updated by $\omega_{ij} = \omega_{ij} + \eta^P \cdot (\delta_i^h)^P \cdot f_j(x, y)$. Similarly, the weight connections between the hidden and output layers are updated using the expression $\omega_{ki}^P = \omega_{ki}^P + \eta^P \cdot (\delta_k^o)^P \cdot H_i^P$.

4. *Classification.* For each pixel vector in the input data cube f, calculate (in parallel) $\sum_{j=1}^{P} O_k^j$, with $k = 1, 2, ..., C$. A classification label for each pixel can be obtained using the winner-take-all criterion commonly used in neural networks by finding the cumulative sum with maximum value, say $\sum_{j=1}^{P} O_{k^*}^j$, with $k^* = \arg\{\max_{1 \le k \le C} \sum_{j=1}^{P} O_k^j\}$.

7.3 Experimental Results

This section provides an assessment of the effectiveness of the parallel algorithms described in the previous section. The section is organized as follows. First, we describe a framework for the assessment of heterogeneous algorithms and provide an overview of the heterogeneous and homogeneous networks used in this work for evaluation purposes. Second, we briefly describe the hyperspectral data set used in the experiments. Performance data are given in the last subsection.

7.3.1 Performance Evaluation Framework

Following a recent study [17], we assess the proposed heterogeneous algorithms using the basic postulate that they cannot be executed on a heterogeneous network faster than its homogeneous prototype on an equivalent homogeneous cluster network. Let us assume that a heterogeneous network consists of $\{p_i\}_i^P$ heterogeneous workstations with different cycle-times w_i, which span m communication segments $\{s_j\}_{j=1}^m$, where $c^{(j)}$ denotes the communication speed of segment s_j. Similarly, let $p^{(j)}$ be the number of processors that belong to s_j, and let $w_t^{(j)}$ be the speed of the t-th processor connected to s_j, where $t = 1, ..., p^{(j)}$. Finally, let $c^{(j,k)}$ be the speed of the communication link between segments s_j and s_k, with $j, k = 1, ..., m$. According to [17], the above network can be considered equivalent to a homogeneous one made up of $\{q_i\}_{i=1}^P$ processors with a constant cycle-time and interconnected through a homogeneous communication network with speed c if, and only if, the following expressions

are satisfied:

$$c = \frac{\sum_{j=1}^{m} c^{(j)} \cdot [\frac{p^{(j)}(p^{(j)}-1)}{2}] + \sum_{j=1}^{m} \sum_{k=j+1}^{m} p^{(j)} \cdot p^{(k)} \cdot c^{(j,k)}}{\frac{P(P-1)}{2}} \tag{7.4}$$

and

$$w = \frac{\sum_{j=1}^{m} \sum_{t=1}^{p^{(j)}} w_t^{(j)}}{P} \tag{7.5}$$

where the first expression states that the average speed of point-to-point communications between processors $\{p_i\}_{i=1}^{P}$ in the heterogeneous network should be equal to the speed of point-to-point communications between processors $\{q_i\}_{i=1}^{P}$ in the homogeneous network, with both networks having the same number of processors. On the other hand, the second expression simply states that the aggregate performance of processors $\{p_i\}_{i=1}^{P}$ should be equal to the aggregate performance of processors $\{q_i\}_{i=1}^{P}$.

We have configured two networks of workstations to serve as sample networks for testing the performance of the proposed heterogeneous hyperspectral imaging algorithm. The networks are considered approximately equivalent under the above framework. Their description follows:

- *Fully heterogeneous network*. This network, already described and used in Chapter 2 of the present volume, consists of 16 different workstations and 4 communication segments, where processors $\{p_i\}_{i=1}^{4}$ are attached to communication segment s_1, processors $\{p_i\}_{i=5}^{8}$ communicate through s_2, processors $\{p_i\}_{i=9}^{10}$ are interconnected via s_3, and processors $\{p_i\}_{i=11}^{16}$ share the communication segment s_4. The communication links between the different segments $\{s_j\}_{j=1}^{4}$ only support serial communication. The communication network of the fully heterogeneous network consists of four relatively fast homogeneous communication segments, interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.
- *Fully homogeneous network*. Consists of 16 identical Linux workstations $\{q_i\}_{i=1}^{16}$ with a processor cycle-time of $w = 0.0131$ seconds per megaflop, interconnected via a homogeneous communication network where the capacity of links is $c = 26.64$ milliseconds.

Finally, in order to test the proposed algorithm on a large-scale parallel platform, we have also experimented with Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center. The system is composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory. The total peak performance of the system is 2457.6 GFlops. Along with the 512-processor computer core, Thunderhead has several nodes attached to the core with 2 Ghz optical fibre Myrinet. In all considered platforms, the operating system used at the time of the

experiments was Linux Fedora Core, and MPICH was the message-passing library used (see http://www-unix.mcs.anl.gov/mpi/mpich).

7.3.2 Hyperspectral Data Sets

Before empirically investigating the performance of the proposed parallel hyperspectral imaging algorithms in the five considered platforms, we first describe the hyperspectral image scene that will be used in the experiments. The scene was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The relatively large area covered (512 lines by 217 samples) results in a total image size of more than 1 GB. Figure 7.3(a) shows the spectral band at 587 nm wavelength and a sub-scene (called hereinafter Salinas A), which comprises 83 × 86 pixels and is dominated by directional features. Figure 7.3(b) shows the ground-truth map, in the form of a class assignment for each labeled pixel with 15 mutually exclusive ground-truth classes. As shown by Figure 7.3(b), ground truth is available for nearly half of the Salinas scene. The data set above represents a very challenging classification problem (due to the spectral similarity of most classes, discriminating among them is very difficult). This fact has made the scene a universal and widely used benchmark to validate the classification accuracy of hyperspectral algorithms [5].

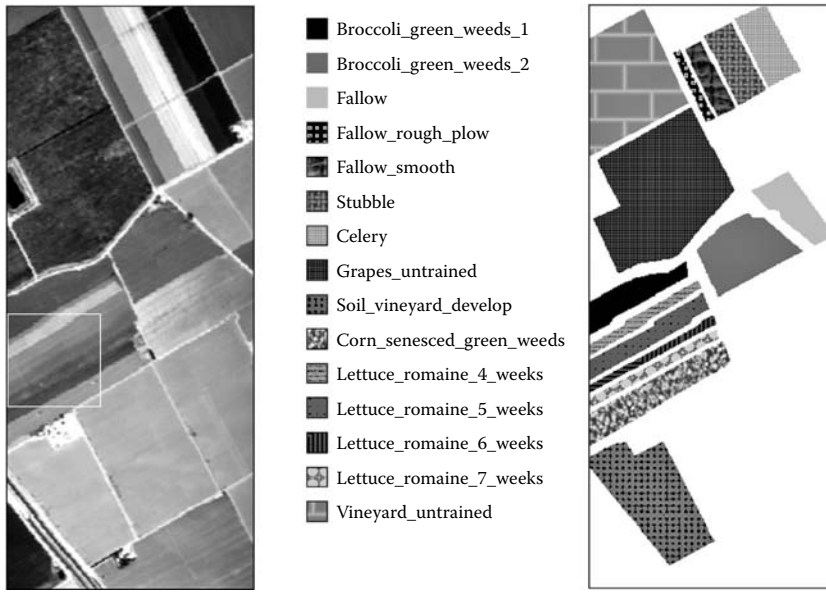

Figure 7.3 AVIRIS scene of Salinas Valley, California (a), and land-cover ground classes (b).

TABLE 7.1 Classification Accuracies (in Percentage) Achieved by The Parallel Neural Classifier for the AVIRIS Salinas Scene Using Morphological Features, PCT-Based Features, and the Original Spectral Information (Processing Times in a Single Thunderhead Node are Given in the Parentheses)

AVIRIS Salinas Class Label	**Spectral Information (2981)**	**PCT-Based Features (3256)**	**Morphological Features (3679)**
Fallow rough plow	96.51	91.90	96.78
Fallow smooth	93.72	93.21	97.63
Stubble	94.71	95.43	98.96
Celery	89.34	94.28	98.03
Grapes untrained	88.02	86.38	95.34
Soil vineyard develop	88.55	84.21	90.45
Corn senesced green weeds	82.46	75.33	87.54
Lettuce romaine 4 weeks	78.86	76.34	83.21
Lettuce romaine 5 weeks	82.14	77.80	91.35
Lettuce romaine 6 weeks	84.53	78.03	88.56
Lettuce romaine 7 weeks	84.85	81.54	86.57
Vineyard untrained	87.14	84.63	92.93
Overall accuracy	87.25	86.21	95.08

In order to test the accuracy of the proposed parallel morphological/neural classifier, a random sample of less than 2% of the pixels was chosen from the known ground-truth of the Salinas scene described above. Morphological profiles were then constructed in parallel for the selected training samples using 10 iterations, which resulted in feature vectors with dimensionality of 20 (i.e., 10 structuring element iterations for the *opening series* and 10 iterations for the *closing series*). The resulting features were then used to train the parallel back-propagation neural network classifier with one hidden layer, where the number of hidden neurons was selected empirically as the square root of the product of the number of input features and information classes (several configurations of the hidden layer were tested and the one that gave the highest overall accuracies was reported). The trained classifier was then applied to the remaining 98% of the labeled pixels in the scene, yielding the classification accuracies shown in Table 7.1.

For comparative purposes, the accuracies obtained using the full spectral information and PCT-reduced features as input to the neural classifier are also reported in Table 7.1. As shown in the table, morphological input features substantially improve individual and overall classification accuracies with regard to PCT-based features and the full spectral information (e.g., for the directional 'lettuce' classes contained in the Salinas A subscene). This is not surprising since morphological operations use both spatial and spectral information as opposed to the other methods, which rely on spectral information alone. For illustrative purposes, Table 7.1 also includes (in the parentheses) the algorithm processing times in seconds for the different approaches tested, measured on a single processor in the Thunderhead system. Experiments were performed using the GNU-C/C++ compiler in its 4.0 version. As shown in table,

TABLE 7.2 Execution Times (in Seconds) and Performance Ratios Reported for the Homogeneous Algorithms Versus The Heterogeneous Ones on the Two Considered Networks

	Homogeneus Network		Heterogeneus Network	
Algorithm	**Time**	**Homo/Hetero**	**Time**	**Homo/Hetero**
HeteroMORPH	221	1.11	206	10.98
HomoMORPH	198		2261	
HeteroCOM	289	1.12	242	11.86
HomoCOM	258		2871	
HeteroNEURAL	141	1.12	130	9.70
HomoNEURAL	125		1261	

the computational cost was slightly higher when morphological feature extraction was used.

7.3.3 Assessment of the Parallel Algorithm

To investigate the properties of the parallel morphological/neural classification algorithm developed in this work, the performance of its two main modules (HeteroMORPH and HeteroNEURAL) was first tested by timing the program using the heterogeneous network and its equivalent homogeneous one. For illustrative purposes, an alternative implementation of HeteroMORPH without 'overlapping scatter' was also tested; i.e., in this implementation the overlap border data are not replicated between adjacent processors but communicated instead. This approach is denoted as HeteroCOM, with its correspondent homogeneous version designated by HomoCOM.

As expected, the execution times reported in Table 7.2 for the three considered heterogeneous algorithms and their respective homogeneous versions indicate that the heterogeneous implementations were able to adapt much better to the heterogeneous computing environment than the homogeneous ones, which were only able to perform satisfactorily on the homogeneous network. For the sake of comparison, Table 7.2 also shows the performance ratios between the heterogeneous algorithms and their respective homogeneous versions (referred to as Homo/Hetero ratio in the table and simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm).

From Table 7.2, one can also see that the heterogeneous algorithms were always several times faster than their homogeneous counterparts in the heterogeneous network, while the homogeneous algorithms only slightly outperformed their heterogeneous counterparts in the homogeneous network. The Homo/Hetero ratios reported in the table for the homogeneous algorithms executed on the homogeneous network were indeed very close to 1, a fact that reveals that the performance of heterogeneous algorithms was almost the same as that evidenced by homogeneous algorithms when they were run in the same homogeneous environment. The above results demonstrate

TABLE 7.3 Communication (COM), Sequential Computation (SEQ), and Parallel Computation (PAR) Times for the Homogeneous Algorithms Versus the Heterogeneous Ones on the Two Considered Networks After Processing the AVIRIS Salinas Hyperspectral Image

	Homogeneous Network			**Heterogeneous Network**		
	COM	SEQ	PAR	COM	SEQ	PAR
HeteroMORPH	7	19	202	11	16	190
HomoMORPH	14	18	180	6	16	2245
HeteroCOM	57	16	193	52	15	182
HomoCOM	64	15	171	69	13	2194
HeteroNEURAL	4	27	114	7	24	106
HomoNEURAL	9	27	98	3	24	1237

the flexibility of the proposed heterogeneous algorithms, which were able to adapt efficiently to the two considered networks.

Interestingly, Table 7.2 also reveals that the performance of the heterogeneous algorithms on the heterogeneous network was almost the same as that evidenced by the equivalent homogeneous algorithms on the homogeneous network (i.e., the algorithms achieved essentially the same speed, but each on its network). This seems to indicate that the heterogeneous algorithms are very close to the optimal heterogeneous modification of the basic homogeneous ones. Finally, although the Homo/Hetero ratios achieved by HeteroMORPH and HeteroCOM are similar, the processing times in Table 7.2 seem to indicate that the data replication strategy adopted by HeteroMORPH is more efficient than the data communication strategy adopted by HeteroCOM in our considered application.

To further explore the above observations in more detail, an in-depth analysis of computation and communication times achieved by the different methods is also highly desirable. For that purpose, Table 7.3 shows the total time spent by the tested algorithms in communications (labeled as COM in the table) and computations in the two considered networks, where two types of computation times were analyzed, namely, sequential (those performed by the root node with no other parallel tasks active in the system, labeled as SEQ in the table) and parallel (the rest of the computations, i.e., those performed by the root node and/or the workers in parallel, labeled as PAR in the table). The latter includes the times in which the workers remain idle. It is important to note that our parallel implementations have been carefully designed to allow overlapping of communications and computations when no data dependencies are involved.

It can be seen from Table 7.3 that the COM scores were very low when compared to the PAR scores in both HeteroMORPH and HeteroNEURAL. This is mainly due to the fact that these algorithms involve only a few inter-processor communications, which leads to almost complete overlapping between computations and communications in most cases. In the case of HeteroMORPH, it can be observed that the SEQ and PAR scores are slightly increased with regard to those obtained for HeteroCOM

TABLE 7.4 Load-Balancing Rates for the Parallel Algorithms on the Homogeneous and Heterogeneous Network

	Homogeneous Network		Heterogeneous Network	
Algorithm	D_{All}	D_{Minus}	D_{All}	D_{Minus}
HeteroMORPH	1.03	1.02	1.05	1.01
HomoMORPH	1.05	1.01	1.59	1.21
HeteroCOM	1.06	1.04	1.09	1.03
HomoCOM	1.07	1.03	1.94	1.52
HeteroNEURAL	1.02	1.01	1.03	1.01
HomoNEURAL	1.03	1.01	1.39	1.19

as a result of the the data replication strategy introduced by the former algorithm. However, Table 7.3 also reveals that the COM scores measured for HeteroCOM were much higher than those reported for HeteroMORPH, and could not be completely overlapped with computations due to the high message traffic resulting from communication of full hyperspectral pixel vectors across the heterogeneous network. This is the main reason why the execution times measured for HeteroCOM were the highest in both networks, as already reported by Table 7.2. Finally, the fact that the PAR scores produced by the homogeneous algorithms executed on the heterogeneous network are so high is likely due to a less efficient workload distribution among the heterogeneous workers. Therefore, a study of load balance is highly required to fully substantiate the parallel properties of the considered algorithms.

In order to measure load balance, Table 7.4 shows the imbalance scores achieved by the parallel algorithms on the two considered networks. The imbalance is defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor runtimes, respectively. Therefore, perfect balance is achieved when $D = 1$. In the table, we display the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. As we can see from Table 7.4, both the HeteroMORPH and HeteroNEURAL algorithms were able to provide values of D_{All} close to 1 in the two considered networks, which indicates that the proposed heterogeneous data partitioning algorithm is effective. Further, the above algorithms provided almost the same results for both D_{All} and D_{Minus} while, for the homogeneous versions, load balance was much better when the root processor was not included. While the homogeneous algorithms executed on the heterogeneous network provided the highest values of D_{All} and D_{Minus} (and hence the highest imbalance), the heterogeneous algorithms executed on the homogeneous network resulted in values of D_{Minus} that were close to optimal.

Despite the fact that conventional feature extraction algorithms (such as those based on PCT) do not take into account the spatial information explicitly into the computations—a fact that has traditionally been perceived as an advantage for the development of parallel implementations—and taking into account that both HeteroMORPH and HeteroNEURAL introduce redundant information expected to slow

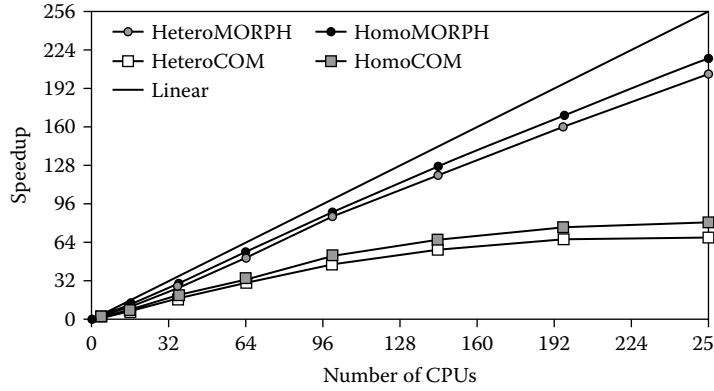

Figure 7.4 Scalability of parallel morphological feature extraction algorithms on Thunderhead.

down the computation a priori, the results in Table 7.4 indicate that the two heterogeneous algorithms are effective in finding an appropriate workload distribution among the heterogeneous processors. On the other hand, the higher imbalance scores measured for HeteroCOM (and its homogeneous version) are likely due to the impact of inter-processor communications. In this case, further research is required to adequately incorporate the properties of the heterogeneous communication network into the design of the heterogeneous algorithm.

Taking into account the results presented above, and with the ultimate goal of exploring issues of scalability (considered to be a highly desirable property in the design of heterogeneous parallel algorithms), we have also compared the performance of the heterogeneous algorithms and their homogeneous versions on the Thunderhead Beowulf cluster. Figure 7.4 plots the speedups achieved by multi-processor runs of the heterogeneous parallel implementations of the morphological feature extraction algorithm over the corresponding single-processor runs of each considered algorithm on Thunderhead. For the sake of comparison, Figure 7.4 also plots the speedups achieved by multi-processor runs of the homogeneous versions on Thunderhead. On the other hand, Figure 7.5 shows similar results for the parallel neural network classifier. As Figure 7.4 and 7.5 show, the scalability of heterogeneous algorithms was essentially the same as that evidenced by their homogeneous versions, with both HeteroNEURAL and HeteroMORPH showing scalability results close to linear in spite of the fact that the two algorithms introduce redundant computations expected to slow down the computation a priori. Quite opposite, Figure 7.4 shows that the speedup plot achieved by HeteroCOM flattens out significantly for a high number of processors, indicating that the ratio of communications to computations is progressively more significant as the number of processors is increased, and parallel performance is significantly degraded. The above results clearly indicate that the proposed data replication strategy is more appropriate than the tested data communication strategy in the design of a

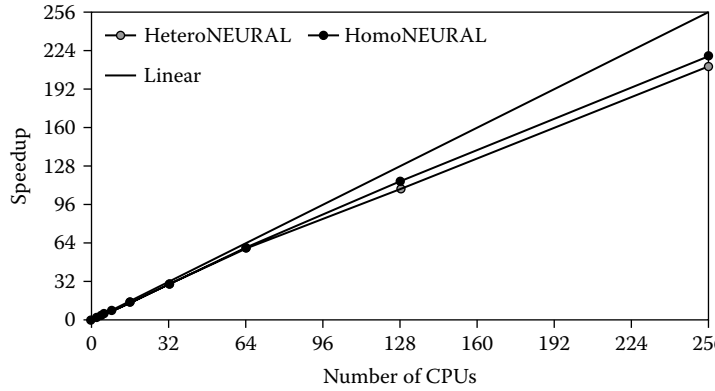

Figure 7.5 Scalability of parallel neural classifier on Thunderhead.

parallel version of morphological feature extraction in the context of remote sensing applications.

Overall, experimental results in our study reveal that the proposed heterogeneous parallel algorithms offer a relatively platform-independent and highly scalable solution in the context of realistic hyperspectral image analysis applications. Contrary to common perception that spatial/spectral feature extraction and back-propagation learning algorithms are too computationally demanding for practical use and/or (near) real-time exploitation in hyperspectral imaging, the results in this chapter demonstrate that such approaches are indeed appealing for parallel implementation, not only because of the regularity of the computations involved in such algorithms, but also because they can greatly benefit from the incorporation of redundant information to reduce sequential computations at the master node and involve minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks.

7.4 Conclusions and Future Research

In this chapter, we have presented an innovative parallel algorithm for hyperspectral image analysis based on morphological neural networks, and implemented several variations of the algorithm on both heterogeneous and homogeneous networks and clusters. The parallel performance evaluation strategy conducted in this work was based on experimentally assessing the heterogeneous algorithm by comparing its efficiency on a fully heterogeneous network (made up of processing units with different speeds and highly heterogeneous communication links) with the efficiency achieved by its equivalent homogeneous version on an equally powerful homogeneous network. Scalability results on a massively parallel commodity cluster are also provided.

Experimental results in this work anticipate that the (readily available) computational power offered by heterogeneous architectures offers an excellent alternative for the efficient implementation of hyperspectral image classification algorithms based on morphological neural networks, which can successfully integrate the spatial and spectral information in the data in simultaneous fashion. In future research, we are planning on implementing the proposed parallel neural algorithm using hardware architectures taking advantage of the efficient systolic array design already conducted by the morphological and neural stages of the algorithm [18].

7.5 Acknowledgment

The authors, thank J. Dorband, J. C. Tilton, and J. A. Gualtieri for their support with experiments on NASA's Thunderhead system. They also acknowledge their appreciation for Profs. M. Valero and F. Tirado.

References

[1] C.-I. Chang. *Hyperspectral imaging: Techniques for spectral detection and classification*. Kluwer: New York, 2003.

[2] R. O. Green. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment,* vol. 65, pp. 227–248, 1998.

[3] G. Aloisio and M. Cafaro. A dynamic earth observation system. *Parallel Computing*, vol. 29, pp. 1357–1362, 2003.

[4] D. A. Landgrebe. *Signal theory methods in multispectral remote sensing*. Wiley: Hoboken, 2003.

[5] A. Plaza, P. Martinez, J. Plaza, and R. M. Perez. Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 466–479, 2005.

[6] T. El-Ghazawi, S. Kaewpijit, and J. L. Moigne. Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality. *Proceedings of the IEEE International Conference on Cluster Computing,* pp. 102–110, 2001.

[7] A. Plaza, D. Valencia, J. Plaza, and P. Martinez. Commodity cluster-based parallel processing of hyperspectral imagery. *Journal of Parallel and Distributed Computing*, vol. 66, pp. 345–358, 2006.

[8] P. Wang, K. Y. Liu, T. Cwik, and R. O. Green. MODTRAN on supercomputers and parallel computers. *Parallel Computing*, vol. 28, pp. 53–64, 2002.

[9] T. Sterling. Cluster computing. *Encyclopedia of Physical Science and Technology,* vol. 3, 2002.

[10] J. Dorband, J. Palencia, and U. Ranawake. Commodity clusters at Goddard Space Flight Center. *Journal of Space Communication,* vol. 3, pp. 227–248, 2003.

[11] A. Lastovetsky. *Parallel computing on heterogeneous networks*. Wiley-Interscience: Hoboken, NJ, 2003.

[12] G. X. Ritter, P. Sussner, and J. L. Diaz. Morphological associative memories. *IEEE Transactions on Neural Networks*, vol. 9, pp. 281–293, 2004.

[13] P. Soille. *Morphological image analysis: Principles and applications*. Springer: Berlin, 2003.

[14] J. Plaza, A. Plaza, R. M. Perez, and P. Martinez. Automated generation of semi-labeled training samples for nonlinear neural network-based abundance estimation in hyperspectral data. *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 345–350, 2005.

[15] S. Suresh, S. N. Omkar, and V. Mani. Parallel implementation of back-propagation algorithm in networks of workstations. *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, pp. 24–34, 2005.

[16] J. Plaza, R. M. Perez, A. Plaza, P. Martinez and D. Valencia. Parallel morphological/neural classification of remote sensing images using fully heterogeneous and homogeneous commodity clusters. *Proceedings of the IEEE International Conference on Cluster Computing*, pp. 328–337, 2006.

[17] A. Lastovetsky and R. Reddy. On performance analysis of heterogeneous parallel algorithms. *Parallel Computing*, vol. 30, pp. 1195–1216, 2004.

[18] D. Zhang and S. K. Pal. *Neural Networks and Systolic Array Design*. World Scientific: Singapore, 2002.