Simulation of Self-Organizing Neural Nets: A Comparison between a Transputer Ring and a Connection Machine CM-2

K. Obermayer, H. Heller, H. Ritter and K. Schulten
Beckman-Institute for Advanced Science and Technology,
University of Illinois at Urbana-Champaign,
405 N. Matthews Av., Urbana, IL 61801

Abstract:

In this contribution we describe parallel algorithms implementing Kohonen's "Self-organizing Feature Maps" on a Transputer systolic ring and on a Connection Machine CM-2. We implemented Kohonen's algorithm with the goal to study its ability for biological modelling and to explain the formation and the adaptive properties of topographic feature maps in the CNS of higher animals. Therefore our attention lies on networks with a high number of units receiving pattern vectors from of a high dimensional input space.

In the following we present benchmark studies (*i*) to measure the performance of the algorithm as a function of important network parameters on both systems, (*ii*) to compare the performance on the MIMD - Transputer systolic ring ("coarse-grained" implementation) with the performance on the SIMD - Connection Machine CM-2 ("fine-grained" implementation), and (*iii*) to measure the speedup of the algorithm depending on the number of physical processors used. The results are supplemented by measurements of communication times between the processors to allow predictions for related algorithms involving higher communication between units. A short overview over results obtained with both parallel machines is given.

1. Introduction:

The development of computer technology and computer science in the last two decades has led not only to a variety of fast and powerful machines, but also to a high flexibility in their use. The concept of the computer as a universal machine, and the fact that the ability to perform a certain task could be transferred from one machine to another by porting the code, supported the notion that the ability to perform a certain task resides in the software independent of a specific hardware configuration. In the last years the limits of these concepts became more and more evident as a growing number of applications called for machines performing typical human tasks (pattern recognition, sensori-motor-control, associative memory) and experience showed that sequential computers using conventional algorithms performed very poorly. Therefore, several new types of algorithms have been proposed in recent years. These, mostly biologically inspired, algorithms ("neural networks") are characterized by a large number of simple, independent computational elements ("neurons"), that exchange data via a large number of connections.

Due to their inherent parallelism neural networks offer a powerful approach for processing large amounts of data, but their full potential can be utilized only if they are implemented on a parallel machine with the appropriate architecture. Therefore, the design of the hardware can no longer be seen as independent from the software, but is to a significant degree coupled to the algorithm and, ideally, both must be optimized together.

One major consideration in this optimization concerns the extent to which the task is divided into parallel subtasks (its "fine-grainedness"). In our contribution we report a study of this issue for the implementation of a class of neural networks known as "self-organizing feature maps" on a Transputer systolic ring and on a Connection Machine CM-2. The algorithm was implemented with the goal to study the formation and the adaptive properties of topographic feature maps in the central nervous system of higher animals within a computationally feasible model. Since one predominant feature of the brain is the large number of massivly interconnected computational elements ($1mm^2$ of cortex consists of 10^5 neurons with 1000 synaptic connections each), the focus of our investigations is directed at networks comprising a large number of neurons, each receiving a high number of external inputs.

In the following we will present results of simulations obtained with both systems, and we will compare the performance of the Transputer systolic ring ("coarse-grained" parallelization) with the Connection-Machine CM-2 ("fine-grained" parallelization) for the self-organizing feature maps.

2. Algorithm:

Self-organizing feature maps (further referred to as SFM) are based on an algorithm proposed by Kohonen [1, 2]. These networks have several properties recognized today as very essential for the architecture of the brain, namely layered and topographic organization of neurons, lateral interaction, Hebb like synaptic plasticity, and capability of unsupervised learning, and these algorithms are fast and efficient during learning as well as during retrieval. SFM's have been applied in the past to a variety of tasks, such as visuomotor control in a camera-robotarm-system [3, 4], modelling the formation of topographic maps in the somatosensory cortex [5], optimization problems [6] and speech-processing [7].

The SFM employ a large number of computational units ("neurons") arranged in one or in multiple layers. Each unit acts as an elementary storage element for a pattern vector of varying complexity. During both learning and retrieval, units receive patterns presented to an array of "receptors" in parallel, competing with each other for their response to the received pattern and for participation in learning adjustments. Each unit increases its responsiveness towards the currently presented pattern using a Hebb-like adjustment rule. The degree of adjustment is a function of the relative match between the pattern vector currently stored at this unit and the new input, and decreases with decreasing similarity between each. The units are interconnected by links specifying a "topology" of the network.This topology coordinates the adjustment steps of different units: units interconnected over short paths, comprising only few links, tend to act in a more similar way. This leads to cooperative effects, which strongly shortens the learning time.

Figure 1 illustrates the version of the model, which we implemented on both parallel machines and on which the following simulations were based. The network architecture was motivated by a particular biological situation, the representation of sensory surfaces (e.g. skin, retina) within certain areas of the cortex of higher animals. The model consists of two layers: a "receptor"-layer and a layer containing the computational units ("neural sheet"). The computational units are arranged as a two-dimensional square lattice, such that each

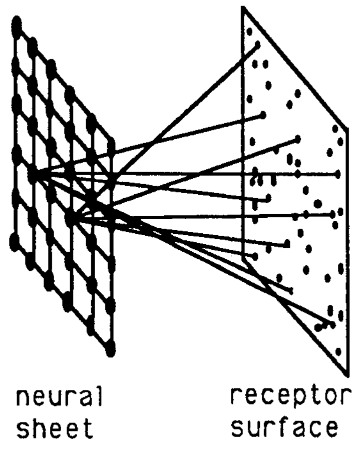


Fig. 1: Implemented version of the SFM-algorithm (explanation see text).

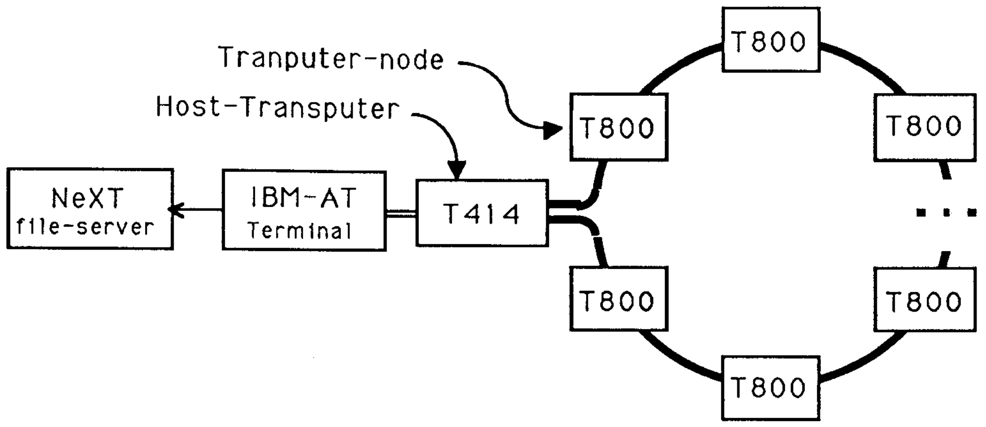


Fig. 2: Schematic diagram of the type of systolic ring network used in our application.

unit has four nearest neighbors (as indicated by the links). Each unit is connected to each "receptor" of the receptor layer via modifyable connections, whose connection strengths can be interpreted as components of a stored "pattern-vector". An input pattern consists of a set of non-zero excitations on the receptor array, represented by an input vector $\vec{r} = \{r_1, r_2, \ldots, r_n\}^T$ ($r_i \geq 0$), where n is the number of the receptors in the receptor surface. In our simulations the input vector $\vec{r}$ is fairly high-dimensional, but the input-patterns are chosen from a low-dimensional manifold, which can be described by a few parameters only. The vector $\vec{r}$ is received by each neuron in parallel, and each neuron, identified by its lattice location (k, l), computes its response o_{kl} as a weighted sum

$$o_{kl} = \sum_i w_{kli} r_i \tag{1}$$

over all receptor outputs r_i. Each coefficient w_{kli} is a measure of the "strength" of the connection from receptor i to neuron (k, l). Following the algorithm of Kohonen the neuron (r, s) with the maximal sum o_{rs} is selected. Then all neuron outputs o_{kl} are replaced by the values $h_{rs;kl}$ of a Gaussian output function centered at the previously selected "neuron" (r, s):

$$h_{rs;kl}(t) = \exp\left[-((r-k)^2 + (s-l)^2)/\sigma_h^2(t)\right] \tag{2}$$

Subsequently, the connection strengths are changed according to a Hebb-type learning rule:

$$w_{kli}(t+1) = (w_{kli}(t) + \epsilon(t) h_{rs;kl}(t) \cdot r_i) / \sqrt{\Sigma_i (w_{kli}(t) + \epsilon(t) h_{rs;kl}(t) \cdot r_i)^2} \tag{3}$$

The learning step width $\epsilon(t)$ decreases linearly from an initial value ϵ_i to a final value ϵ_f, and The width $\sigma_h(t)$ of the output-function $h_{rs;kl}$ decreases linearly during the simulation from an initial value σ_i to a final value σ_f to allow the neurons to gradually specialize for different subregions of the input space.

3. Implementation on the Transputer Systolic Ring

The Transputer systolic ring [1] (Fig. 2) can be equipped with up to 60 nodes [8]. The ring has an outlet to a host T414 Transputer. Each node consists of a Transputer IMS T800G20S running at 20 MHz clock speed, 4 MByte local memory and a parity check logic, which ensures correctness of data stored in the local RAM. The host Transputer

[1]developed by H. Grubmüller, H. Heller and K. Schulten [8]

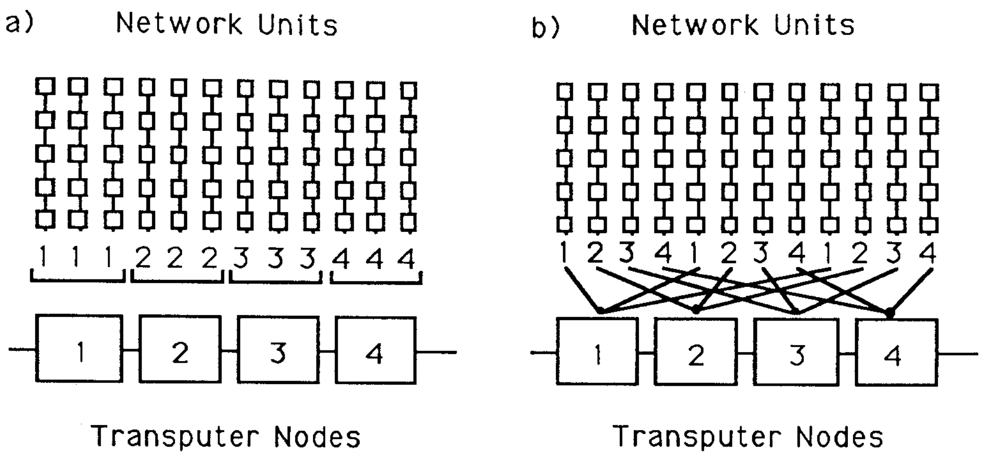


Fig. 3: Two strategies for mapping the network onto the nodes of the Transputer systolic ring.

T414 has 4MByte RAM local memory and serves as a gateway to a PC-AT front end and to a NeXT file server. The implementation was carried out in Occam II using the INMOS TDS2 environment. A more detailed description of the hardware is given in another contribution of these proceedings.[1]

The much more limited number of processors, compared to the Connection Machine, requires a partitioning of the network into units significantly larger than neurons. The increased computational load per unit is, however, partly compensated by the higher performance of a single Transputer, the larger local memory, and the resulting possibility to carry out a larger fraction of communication within a single node.

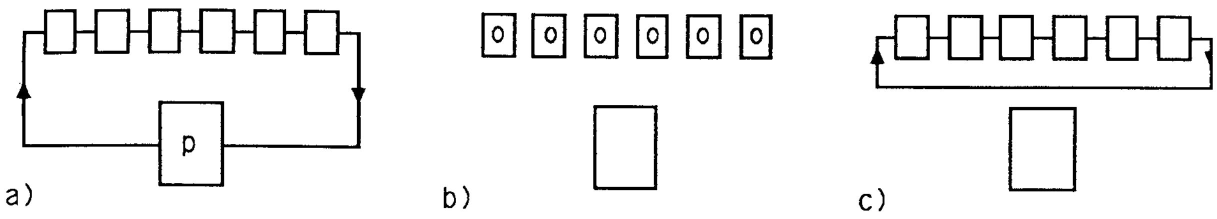
Figure 3 shows two different implementation strategies for the SFM on the Transputer systolic ring. In the first implementation (Fig. 3a, referred to as I1) the network is partitioned into N_p "stripes", where N_p is the number of processors in the ring. These "stripes" are then mapped onto the processors of the ring as depicted in Fig. 3a. In the second implementation (Fig. 3b, referred to as I2), which was proposed by [9], the network is partitioned into $N_s = N_c/N_p$ "stripes" (N_c denotes the number of rows in the network). The neurons are mapped onto the nodes of the ring, such that each processor contains one row of neurons from each "stripe". In this implementation neighboring rows are still mapped onto neighboring nodes, but the distance between rows mapped onto the same processor is maximized.

For an excitation function $h_{rs;kl}$ with finite width the implementation I2 leads to a higher percentage of load balancing than implementation I1, since load balancing is 100% until the width of the excitation function falls below the value N_p. Therefore, one would expect the implementation I2 to be most effective in the "coarse-grained" limit $N_p \ll N_c$, but implementation I2 should generally be preferred for "pure" SOF applications because of its better performance for short ranged excitation functions $h_{rs;kl}$.

The mapping between network units and Transputer nodes shown in Fig. 3a is optimized for local communication between units of the network and should be used, if the algorithm (1) - (3) is to be extended to either multilayer networks or to explicitly include lateral interactions within the "neural sheet" [10].

At the beginning of each adaptation step the parameters describing the current input-pattern are generated by the host-transputer and transmitted along the ring, so that each node can compute a local copy of the input-vector locally (Fig. 4). The neuron

[1]Boehncke K., Heller H. and Schulten K., Molecular Dynamics on a Systolic Ring of Transputers



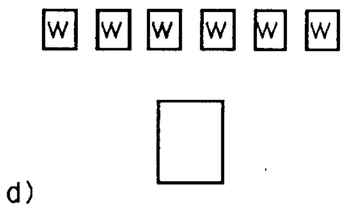


Fig. 4: Schematic diagram of the implementation of the SFM on the Transputer system. The large and small rectangles denote the host Transputer and the Transputer nodes respectively; communication of data is indicated by arrows. **a)** generation and communication of the current pattern p, **b)** computation of the output values o, **c)** determination and communication of the coordinates of the maximally active unit, and **d)** adjustment of link strengths w.

outputs are then computed in parallel and each processor determines the position of the maximally active cell of its "local units". The coordinates and the output-values of all "local" maximally active units are shifted once through the whole ring, so that each processor can determine the coordinates of the "global" maximally active cell. The subsequent adaptation step (3) is carried out simultaneously by all processors in the ring. At fixed intervals, an auxiliary process running parallel with the simulation code evaluates the state of the network and saves the resulting data to disk.

4. Implementation on the Connection Machine CM-2

The Connection Machine is a massively parallel SIMD computer with a large number of very simple, single-bit-processors [11]. The system available at our site consists of 32,768 processors with 8 kByte local memory each, connected in a hypercube-topology, and 1024 WEITEK floating point accelerators which endow the machine with a theoretical performance of 2.5 GFlops. Two frame-buffer-systems allow parallel graphics output of data stored in the local processor memories. A parallel mass-storage system ("Data-Vault") facilitates fast storage and retrieval of the large amount of data describing the evolution of the neural network (up to 6 GByte/run for the simulations presented in this paper). The CM-2 can be accessed from two front end computers (VAX and SUN/4) and parallel CM-commands can be used from within ordinary, sequential code running on one of the front ends.

In view of the large number of available processors, the most natural implementation of a neural network algorithm on the CM-2 is to allocate one processor to one neuron ("fine-grained" approach). If the number of neurons exceeds the number of CM-processors, each physical processor can be "split" into a (for all processors equal) number of "virtual processors" with, however, a proportionally reduced local memory. The processors can be configured as a two-dimensional square lattice to match the geometry of the "neural sheet" and the "receptor"-surface (Fig. 1).

At the beginning of each adaptation step the parameters describing the current stimulus are generated by the front end (Fig. 5a) and then transmitted to all CM-processors, since in our implementation each processor was also used to perform computations for one pixel of the receptor surface. The excitation values of the receptors are computed in parallel. The excitation values are retrieved sequentially by the front end and sent to all

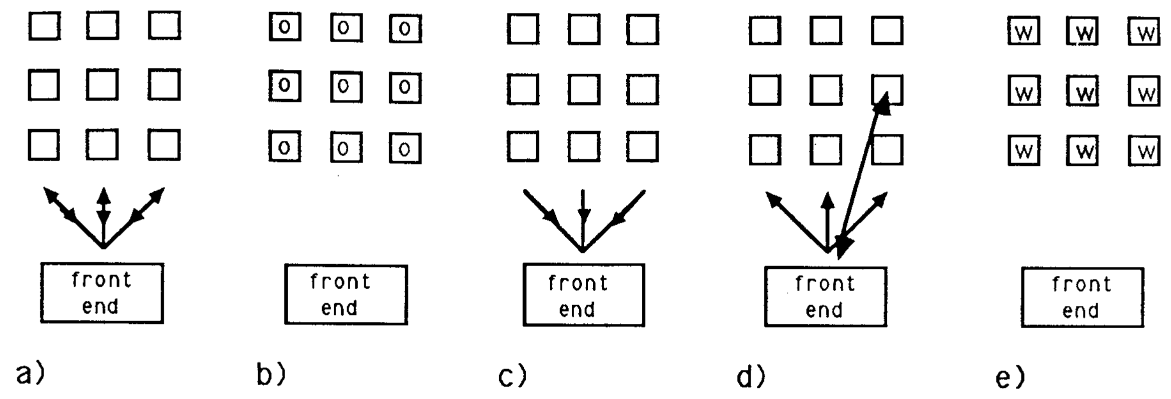


Fig. 5: Schematic diagram of the implementation of the SFM on the CM-2. The small rectangles denote the processors of the CM-2; communication of data is indicated by arrows. **a)** generation and communication of the current pattern, **b)** computation of output values o, **c)** determination of the maximally active cell, **d)** retrieving and broadcasting the coordinates of the maximally active cell, and **e)** adjustment of link strengths w.

processors of the machine (Fig. 5a), which subsequently compute the output values of the units in the network. (Fig. 5b). The maximal output is determined and transmitted to the front end (Fig. 5c), which retrieves the coordinates of the maximally active unit and transmits the coordinates back to the CM (Fig. 5d). The adaptation step (3) is again performed in parallel (Fig. 5e). After a certain number of steps the state of the network is evaluated and the resulting data are stored on the parallel mass storage "Data-Vault".

One main limitation on the realizable network structures is set by the local 8kByte memory, which must accommodate all weight values, together with pointer information about the partner neuron of each weight. A second limitation concerns communication: two or more processors (neurons) cannot simultaneously communicate in parallel with overlapping sets of processors. However, in the Kohonen "short-cut" algorithm the steps involving communication among the units is the maximum operation among the o_{kl} and the communication of the coordinates of the maximally active neuron to all processors. Both steps are implemented as communication operations via the front end machine rather than direct interprocessor communications. All other computation steps are "private" to each neuron and are, therefore, not affected by communication issues. The configuration of the machine as a two-dimensional lattice was chosen only to facilitate the visualization of the network parameters on the frame-buffers of the CM-2 during simulation. Since no interprocessor communication routines are involved, the configuration of the machine in a particular geometry has no effect on the performance of the algorithm. This is, of course, no longer true, if lateral connections within the "neural sheet" are included in an extended version of the SFM-algorithm [10].

5. Performance of the Algorithm

In the following section the performance of the Transputer systolic ring and the CM-2 is measured by the time needed to run parts of or the whole application program. The internal timer of the host-Transputer was used to measure the computation time on the Transputer system. The values given below are the total time-intervals starting after the program has been distributed over all nodes in the ring and ending after the task has been completed. They do not include the time needed to retrieve and process the

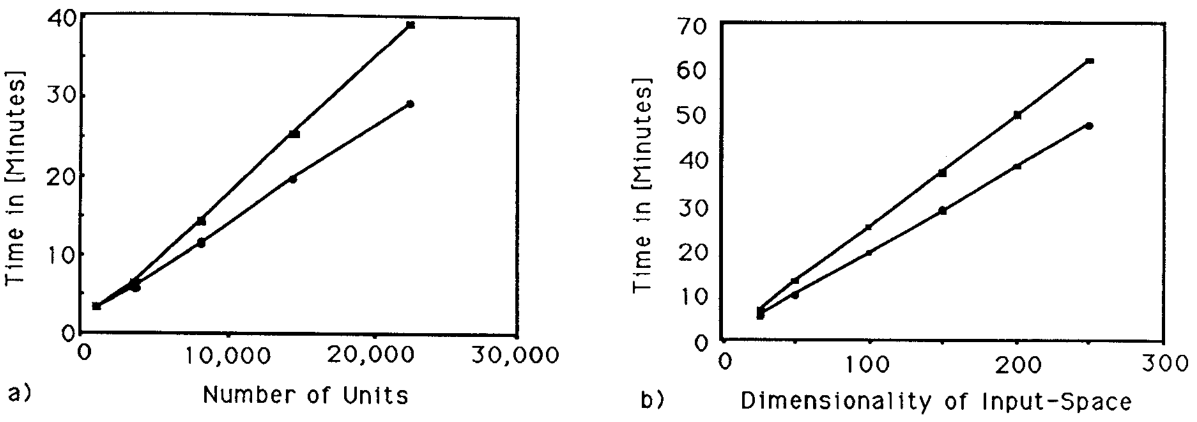


Fig. 6: Performance of the Transputer systolic ring consisting of 30 nodes for the implementations I1 (upper curves) and I2 (lower curves). **a)** Total time for 2,500 iterations as a function of the number of units. The dimensionality of the input-space was 100. **b)** Total time for 2,500 iterations as a function of the dimensionality of the input-space. The network contained 14,400 units. σ_h was decreased linearly from $\sigma_h(0) = 80$ to $\sigma_h(2,500) = 0$ for both performance tests.

network parameters and to store them on the file-server. The time used on the Connection Machine system was measured by the routine CM:time. This routine reports the total ("wall-clock") time, which includes the sequential code sections running on the SUN/4 front end, as well as the time used on the Connection Machine itself. To indicate the distribution of time between front end and CM, the CM-utilization as well as the time used by the CM-processors are reported. It turned out, that the total time measured by CM:time depends strongly on the type of front end (For our code it was approx. 3-4 times longer for the VAX as a front end) and that it is also influenced by other programs running on the front end at the same time. Therefore the values given below were obtained solely for the SUN/4 front end and a front end utilization of more than 97%.

Figure 6 shows the total time required for 5,000 adaptive steps as a function of the number of units (Fig. 6a) and the dimensionality of the input space (Fig. 6b) for the implementations I1 and I2 on the Transputer ring. In both cases the computation time increases linearly with the computational load imposed by the high-dimensional input. If the number of network units is increased, implementation I2 shows improved performance compared to I1, which reflects the better load balancing for the "coarse-grained" mapping.

Figures 7a/8a and 7b/8b are the corresponding diagrams for the implementation on the Connection Machine. On the CM-2 the used number precision can be freely selected, and Figs. 7, 8 show the results for 32-bit and 16- / 64-bit floating point numbers respectively. A comparison shows, that the total time does not decrease, if the floating point precision is reduced from 32 bit to 16 bit. Instead it increases drastically, because for precisions other than 32 bit the floating point operations can no longer be executed by the WEITEK FPU's but have to be calculated by the much slower 1-bit processors.

The total time on the CM-2 is a step-function of the network size, because the number N_p of virtual processors must be an integer multiple of the physical machine size, which is currently limited to multiples of 8K. If the number of "neurons" N_n does not match the number of virtual processors ($N_n \neq N_p$) idle processors lead to an unbalanced computational load. Best performance is obtained for $N_n = N_p$. The height of the steps is not equal for smaller numbers of units, since the fraction of time used on the CM-2 increases relative to the time used on the front end machines. Figure 9 shows the relative performance s (ratio of time used on the CM-2 and time used on the Transputer) of the

Figure 7

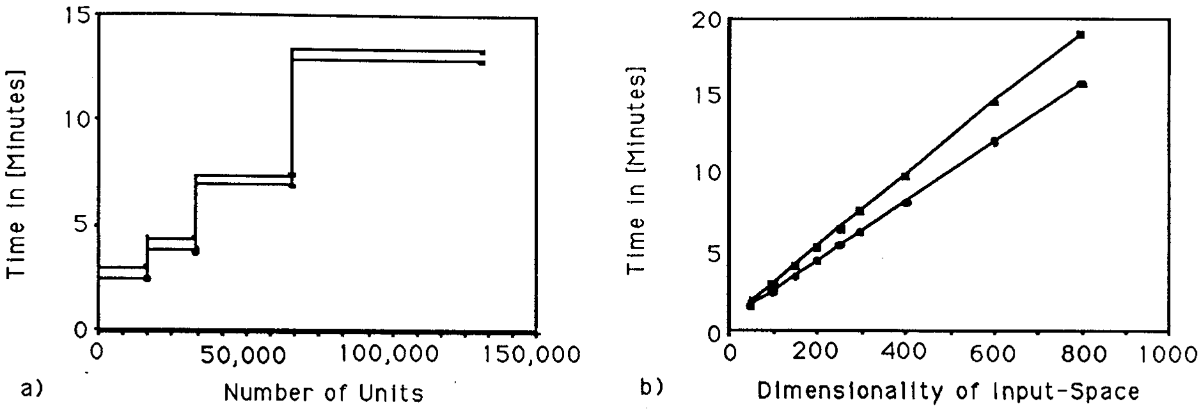


Figure 8

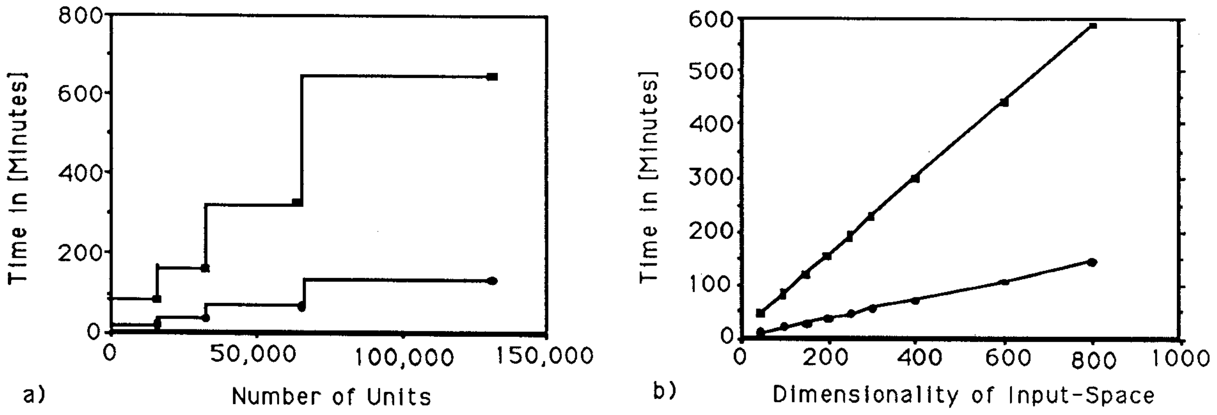


Fig. 7: Performance of a 16K section of the CM-2 with SUN/4 front end for a data precision of 32 bit (IEEE single precision format). **a)** Total time (upper curve) and time used on the processors of the CM-2 (lower curve) for 5,000 iterations as a function of the number of units in the network. The input-dimensionality was 100. The CM-utilization increased from 82% (for 16K units) to 89%, 93% and 96% (for 128K units). **b)** Total time (upper curve) and time used on the processors of the CM-2 (lower curve) for 5,000 iterations as a function of the dimensionality of the input space. The network contained 16,384 units. CM-utilization was 83%.

Fig. 8: Performance of a 16K section of the CM-2 with SUN/4 front end for a data precision of 16 bit (lower curve) and 64 bit (upper curve). **a)** Total time for 5,000 iterations as a function of the number of units in the network. The input-dimensionality was 100. The CM-utilization increased from 94% (for 16K units) to 96%, 98% and 99% (for 128K units) for the 16-bit precision. CM-utilization for the 64-bit precision was 99%. **b)** Total time for 5,000 iterations as a function of the dimensionality of the input space. The network contained 16,384 units. CM-utilization was 94% and 99% respectively. The 64-bit data format used in this performance test did not conform with the IEEE-standard for significant and exponent length, so that the floating point operations were executed by the 1-bit processors.

Transputer system compared to the CM-2 as a function of the number of Transputers for the implementations I1 and I2. The relative performance for implementation I1 is a linear function of the number of nodes in the ring. Implementation I2 generally performes better, especially in the "coarse-grained" limit of a small number of Transputers. In the "fine-grained" limit (i.e. one row per processor) both implementations perform equally well. The improvement in the "coarse-grained" limit depends strongly on the shrinking of the excitation function with time. All results stated above were obtained using a linear decrease of the width $\sigma_h(t)$ from an initial value σ_f to a final value of zero. We found this adaptation scheme appropriate for the application of SFM's to biological modelling. For other applications, e.g. robot control, which require a longer "refinement phase" in the adaptation process, with σ_h having intermediate or small values, we expect a further performance improvement for implementation I2.

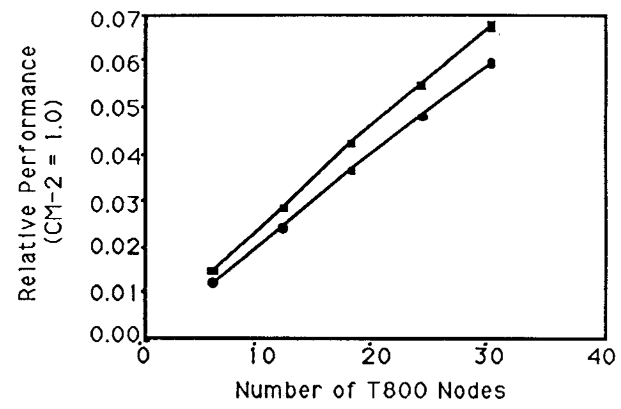


Fig. 9: Relative performance of the I1- (lower curve) and I2-implementation (upper curve) on the Transputer ring compared to the implementation on a 16K section of the CM-2 with SUN/4 front end. The network size was 14,400 units.

An extrapolation of Fig. 9 to $s = 1$ gives a total of 510 Transputer nodes being equivalent to a 16K section of the CM-2 (for 32 bit precision). The number of Transputers may at first sight appear high, but it must be kept in mind that the 16K processors in the CM-2 are supported by 512 32-bit WEITEK FPUs, which make an essential contribution to the overall performance of the CM-2 for the given floating point intensive task. Without the FPU's 39 Transputer nodes give the same performance for the SFM algorithm than the 16K section of the CM-2 assuming 32-bit precision.

In the following we will provide some benchmark results concerning interprocessor communication rates, which might be useful for estimating the relative performance of the CM-2 vs. the Transputer ring for communication intensive neural network algorithms. We define one elementary communication step as the task to communicate the output of each unit (a 32 bit floating point number) to every other unit in the network.

The configuration of the CM-processors as a two-dimensional square lattice with periodic boundary conditions allowed to use the fast NEWS-operations for interprocessor communication. During a first step all data were "rotated" along the X-axis of the processor grid until each processor contained the output-values of all processors with the same X-coordinate. In a second step the collected output-values were "chunked" into fields, which could "rotated" along the Y-axis of the grid with single NEWS-commands.

Figure 10 shows a comparison between the time needed for 1000 communication steps on the CM-2 and on the Transputer system for a neural network consisting of 32,768

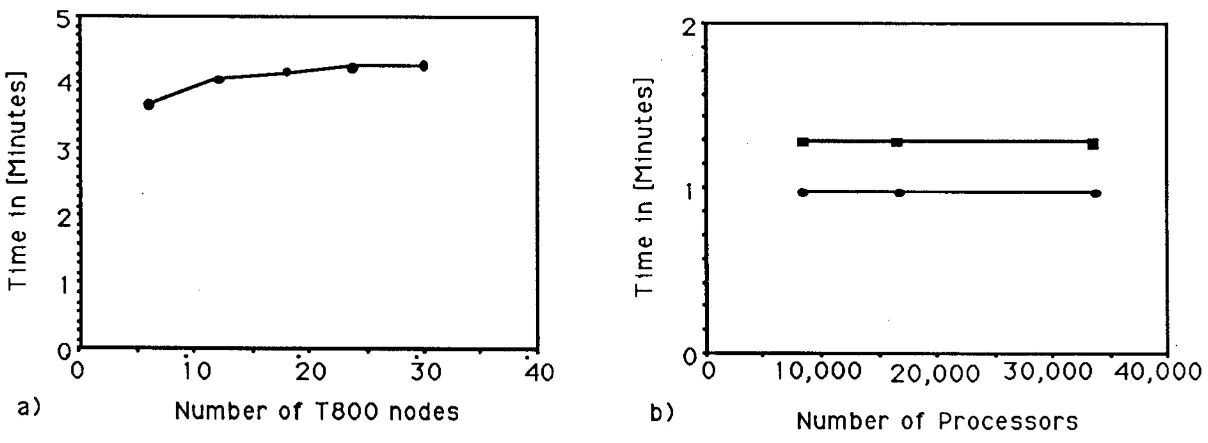


Fig. 10: Total time for 1,000 communication steps as a function of the number of Transputer nodes (**a**) and the number of processors of the CM-2 (**b**). Network size was 32,768 units.

neurons. The slight increase in communication time with the number of Transputer nodes (Fig. 10a) is due to the fact, that for a small number of nodes data can be transmitted in larger chunks. Note, that the time used for one communication step on the CM is considerable and only slightly less than the time used on the Transputer system. For a network with a 100-dimensional input space one communication step on the CM takes approximately as much time as one complete adaptation step of the SFM algorithm. If the "chunking" of data on the CM-2 cannot be done or if the general router commands must be used, communication time on the CM may increase by a factor of 100 or more. Therefore the performance of the Transputer-system will strongly improve relative to the CM-2, if the number of communication operations is comparable to or larger than the number of arithmetic and relational instructions or larger.

6. Scientific Applications:

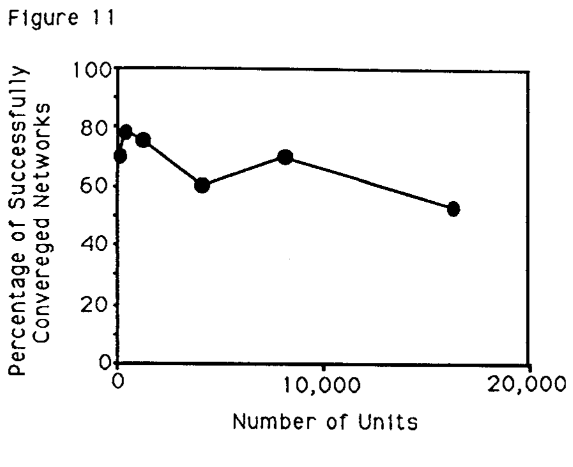
In this section we want to give examples of results we have obtained with both parallel machines and for which the high performance of the parallel hardware was essential. The presented results shall merely serve as an illustration of what aspects of the SFM's can be investigated using parallel supercomputers. If the reader is interested in details, he may wish to refer to [5, 10, 12].

A. Scaling Issues:

Application of the SFM's have so far been restricted to small networks consisting of typically 1000 units or less and data-spaces of at most a few dozen dimensions. Biological systems, on the other hand, are characterized by a large number of highly interconnected neurons and even the smallest functional unit beyond the neuron in the cortex, the "functional column", consists of 10^5 elements. It is clearly necessary to narrow the gap in scale between the current models and actual brain structures, and modern parallel computers begin to be able to simulate reasonable large networks in affordable time. An important point is to evaluate statistically the impact of the scaling-up of the system on its adaptive capabilities, in particular speed of convergence, tolerance against noise and robustness against unfavorable or random initialization. Figure 11 shows one result of Monte-Carlo simulations addressing an important scaling issue: They demonstrate, that the percentage of properly formed feature maps does not change in the "continuum limit": $N_c \to \infty$, $\sigma_h(0)/N_c = const$, i.e. that the number of iterations of eq. (1) - (3) necessary to guarantee a certain percentage of converged maps is independent of the number of neurons.

B. Formation and Readaptation Properties of the Somatotopic Map:

In the somatosensory cortex the complete body surface is represented by a distorted but topographic "map", such that neighboring tactile stimuli on the skin surface excite neighboring neurons in the cortical map. The map is dynamically maintained and changes, driven by stimulation of the skin receptors [13, 14]. The self-organized formation and the readaptation properties of the map of the inner hand-surface have been studied within a large scale model using the SFM algorithm. Two of the artificial maps obtained in the simulations are shown in Fig. 13. The left image shows the representation of the hand surface within the "model-cortex" as it has emerged after 10,000 tactile stimuli. The right image shows a readapted map of the hand-surface after the fourth finger has been "amputated". The units in the area of the "model-cortex", which were deprived of their



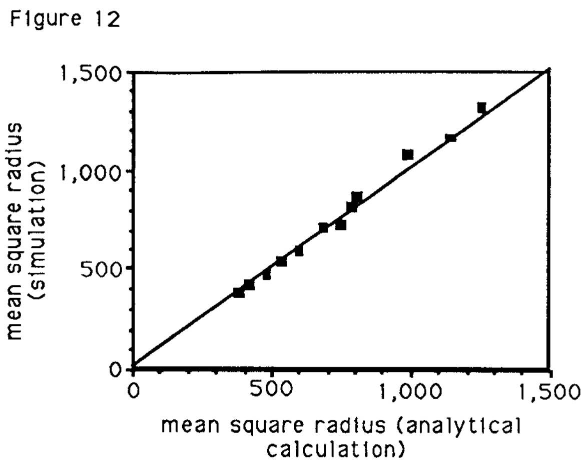


Fig. 11: Percentage of successfully converged SFMs as a function of the number of units in the network. The diagram summarizes the convergence statistics of 240 networks (40 for each data point). The simulations were done on the CM-2 as well as on the Transputer system. The parameters were: $\sigma_h(0)/N_c = 0.625$, $\epsilon = 0.08$, $\sigma_p = 0.2$, 100 input dimensions, and 10,000 iterations.

Fig. 12: A comparison between the analytically calculated radius of the receptive fields of units in a SFM-network with simulation results obtained with the Transputer system.

external input immediately after "amputation", did not end up useless but were allocated to the tactile surface of the neighboring fingers.

C. Formation of Receptive Fields:

Many neurons within the cortex respond to excitations from a certain area of the receptor surface, which is called their *receptive field.* For the formation of a topographic representation it is essential, that the cells develop localized receptive fields. This issue was investigated for the SFM-algorithm analytically as well as by computer simulations. While it is immediately clear from the algorithm, that each receptive field must correspond to some small volume in the high-dimensional input space this does not yet imply that this volume actually will correspond to receptive fields that are also *spatially localized in the two-dimensional receptor surface itself.* Analytical calculations (for details see

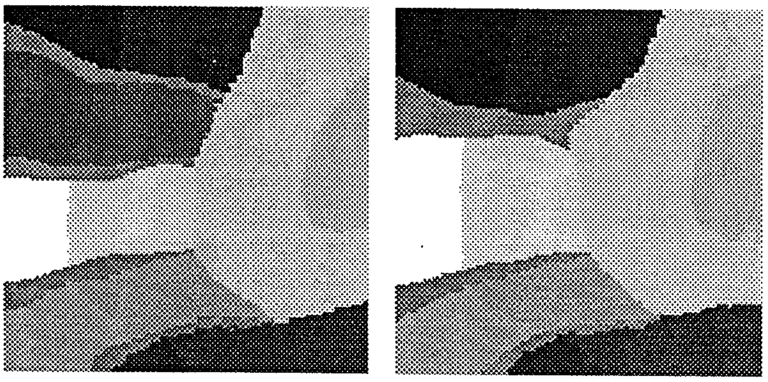


Fig. 13: **a)** Topographic representation of the inner hand surface within a "model-cortex" consisting of 16,384 units arranged in a 128 × 128 square lattice. Each unit was connected to the 800 "tactile receptors" located on the "receptor surface", the "model-skin". The regions representing the palm and the five fingers are are marked by different shadings. **b)** Topographic representation of the inner hand surface after one finger was "amputated". The region formerly corresponding to the "amputated" finger was "reused" to represent part of the receptor surface of the neighboring fingers with a higher resolution. Both maps were generated by the SFM-algorithm implemented on the CM-2.

[12]), as well as computer-simulations confirmed, that all cells develop spatially localized receptive fields. Figure 12 shows a comparison between analytically predicted receptive field diameters and results of a simulation. The correspondence is very accurate, except for parameter values leading to very large receptive fields, for which edge effects of the model become noticeable.

7. Summary:

The goal of this paper was to present a comparison between the performance of a Transputer system and a Connection Machine CM-2 for implementations of an important neural network algorithm, the self-organizing feature maps (SFM). The results show that a Transputer system with about 500 nodes is equivalent to a 16K section of the CM-2, which comes close to the theoretical equivalent of 550 nodes expected from the product data sheet (1.25 GFlops and 2.25 MFlops for a 16K section of the CM-2 and for one Transputer node, respectively). Because of the extremely low communication requirements of the SFM, this algorithm is very well suited for a "fine-grained" implementation as offered by the CM-2 and the performance is only slightly affected by the configuration of both systems.

Since one communication step (involving the communication of one 32-bit floating point number from each neuron to every other neuron) takes at maximum threefold time on the Transputer system, compared to the CM-2, the performance of a Transputer based system will strongly improve for networks algorithms with higher communication requirements.

Acknowledgements: This research has been supported by the University of Illinois at Urbana-Champaign. Computer time on the Connection Machine CM-2 has been made available by the National Center for Supercomputer Applications. The authors would like to thank the Boehringer-Ingelheim Fonds for support of this project by a scholarship to K. Obermayer and R. Kufrin and J. Quinn for their help and support in all technical matters concerning the use of the Connection Machine system.

8. References:

[1] Kohonen T. (1982a), Biol. Cybern. 43:59
[2] Kohonen T. (1982b), Biol. Cybern. 44:135
[3] Martinetz T., Ritter H. and Schulten K. (1989), Proceedings of the International Joint Conference on Neural Networks, Washington 1989, II:351
[4] Ritter H., Martinetz T. and Schulten K. (1989), Neural Networks 2, pp.159
[5] Obermayer K., Ritter H. and Schulten K. (1990), Proceedings of the ICNC-Conference, Düsseldorf 1990, in press
[6] Hueter G.J. (1988), Proceedings of the IEEE International Conference on Neural Networks, Vol. I, p. 85
[7] Kohonen T. (1989), in: I. Aleksander (Ed.), Neural Computing, Kogan Page Ltd., London
[8] Grubmüller H., Heller H. and Schulten K. (1990), Mol. Simulation, in press
[9] Hogdes R.H., Wu C.H. and Wang C.J. (1990), Proceedings of the IJCNN Conference, Washington '90, Vol. II, p. 141
[10] Obermayer K., Ritter H. and Schulten K. (1990), Parallel Computation, in press
[11] Hillis W.D. (1985), The Connection Machine, MIT Press, Cambridge, Mass.
[12] Obermayer K., Ritter H. and Schulten K. (1990), IJCNN Conference Proceedings, San Diego '90, submitted
[13] Merzenich M.M. et al.(1983), Neurosci. 10, 3:639
[14] Merzenich M.M. et al.(1984), J. Comp. Neu. 224:591