Preprint of an article in Neural Networks 12(1): 127–134 (1999),

On the Performance of Neuronal Matching Algorithms

Rolf P. Würtz[a*], Wolfgang Konen[b] Kay-Ole Behrmann

[a] *Institut für Neuroinformatik, Ruhr-Universität Bochum, D-44780 Bochum, Germany*
[b] *Zentrum für Neuroinformatik GmbH, Bochum, Germany*

Abstract

For a solution of the visual correspondence problem we have modified the Self Organizing Map (SOM) to map image planes onto another in a neighborhood- and feature-preserving way. We have investigated the convergence speed of this SOM and Dynamic Link Matching (DLM) on a benchmark problem for the solution of which both algorithms are good candidates. We show that even after careful parameter adjustment the SOM needs a large number of simple update steps and DLM a small number of complicated ones. The results are consistent with an an exponential vs. polynomial scaling behavior with increased pattern size. Finally, we present and motivate a rule for adjusting the parameters of DLM for all problem sizes, which we could not find for SOM.

Keywords:

Correspondence problem, Self-organizing map, Dynamic link matching, Symmetry recognition, Convergence speed, Complexity

1 Introduction

For visual perception in a biological or artificial system the *visual correspondence problem* is of central importance: "Given two images of the same physical object decide which point pairs belong to the same point on the object." This is an special case of pattern matching and usually only considered in constrained form for stereo matching (epipolar constraint) or tracking (small time steps and continuous motion between images). We feel that a generic solution to the *unconstrained* problem at least greatly alleviates many of the difficulties encountered by computer vision research on the one hand and by attempts at modeling human vision on a neuronal level on the other hand. Invariant object recognition, e.g., becomes easy if a correspondence map of sufficient density and reliability can be constructed between objects and stored prototypes (Würtz, 1995; Wiskott, 1996; Wiskott and von der Malsburg, 1996; Würtz, 1997). The study (Konen and von der Malsburg, 1993) is another illustration of the power of such an algorithm. It describes a neural network based on Dynamic Link Matching (DLM) that learns to evaluate input patterns for the presence of one out of three mirror symmetries. That problem might be considered academic but the proposed system compared very favorably with a standard Boltzmann machine (Sejnowski et al., 1986) in terms of training speed. While the Boltzmann machine needed some 10^4 training patterns to classify the symmetries the DLM system with its inherent capability of finding second-order correlations could do with single examples. The basic idea of DLM dates back to (Willshaw and von der Malsburg, 1976).

For a solution of the correspondence problem a mapping $\mathcal{M}$ from one image plane to another one, each carrying features $f(\cdot)$ has to be established, which must satisfy a combination of the following three constraints:

*Corresponding author. Tel.: +49 234 700-7994; Fax: +49 234 7094-210; Email: Rolf.Wuertz@neuroinformatik.ruhr-uni-bochum.de

M1: $\mathcal{M}$ is an one-to-one function from one image plane onto another image plane.

M2: $\mathcal{M}$ is a homeomorphism, i.e. continuous in both directions.

M3: $\mathcal{M}$ is feature-preserving, i.e. $f(\mathcal{M}(x)) = f(x)$.

For application to real-world problems, these constraints have to be relaxed. **M2** has no meaning for discrete spaces, so the somehow ill-defined term of *neighborhood preservation* must take the place of continuity (for a thorough discussion see (Goodhill et al., 1995)). For real images, strict equality of the features cannot be expected, so **M3** is replaced by the constraint that the sum of all local feature similarities should be as large as possible.

There is an abundance of technical solutions for special cases of the correspondence problem, e.g. optical flow or disparity algorithms. Matching algorithms for recognition like in (Lades et al., 1993; Würtz, 1997) also center on special aspects like translations and small distortions and would not apply to large rotations or mirror images without modification. For technical systems, this tailoring to the problem at hand is, of course, the only key to success. When looking for *generic* solutions, a good candidate to solve the correspondence problem is a self-organizing process that develops from an unordered initial state to a mapping that fulfills **M1** through **M3** as good as possible. The Self-organizing Map (SOM) algorithm (Kohonen, 1982; Kohonen, 1990; Kohonen, 1997) seems to be a natural choice. In (Bellando and Kothari, 1996), e.g., it is used to find frame-to-frame correspondences for tracking. It strictly enforces uniqueness (**M1**) in *one* direction, namely that every neuron has only one pointer to an input vector.

Another possibility is Dynamic Link Matching (DLM) (Konen et al., 1994), which also relaxes **M1** such that there is in principle full connectivity between the planes, and the correct correspondences grow, while all others decline to small values. In (Würtz, 1995; Wiskott, 1996; Wiskott and von der Malsburg, 1996) dynamic link matching is used to construct correspondence maps for face recognition.

As self-organization is a notoriously slow process the *convergence speed* is an important detail. Its importance may be lower in the typical application areas that are models of brain development, but for vision applications where correspondences must be established in some milliseconds, it becomes crucial. The authors of (Bellando and Kothari, 1996) give no figures about the computational cost of their system. Short of sound analytical results about convergence times we have used the problem from (Konen and von der Malsburg, 1993) as a benchmark to evaluate the relative performance of DLM (Konen et al., 1994) and the Self Organizing Map (SOM) algorithm (Kohonen, 1982; Kohonen, 1990). In the context of neuronal modeling of vision it should be noted that both algorithms have an inherently sequential component, and consequently the convergence time matters even if the hardware is highly parallel.

The discretised problem is as follows. There are two $N \times N$ layers of neurons X and Y, each neuron carries one of F different features. The algorithm must find a feature-preserving one-one mapping. In the following sections we will define a benchmark problem, present a slight modification of the SOM that includes the feature similarity into the learning rule, then describe DLM, and run both algorithms measuring the performance.

This paper does certainly not attempt a comprehensive comparison between all applications of SOM and the (much fewer) applications of DLM which have been developed during the last decades. Especially, the well-known dimension reduction capabilities of SOM are not considered here. We concentrate on the convergence behavior of SOM and DLM w.r.t. a very specific task, namely the correspondence problem, as described above. Consequently, our results do not necessarily generalize to different tasks which could be tackled by both algorithms. Nevertheless, we believe that the results hold for two-dimensional pattern matching.

It might be argued that the instantiation of the correspondence problem is too simple. Our concern, however, has not been the absolute difficulty of the problem but the relative performance of both algorithms, and that objection would only be serious if there was reason to believe that one of the algorithms would change its behavior qualitatively when moving to real-world images. Furthermore, a very similar problem has proven to be difficult for Boltzmann machines (Sejnowski et al., 1986, see above).

2 Definition of a benchmark problem

A fair comparison of algorithms that were developed to solve different problems and whose full range of applicability is still subject of intensive research is not easy. The least one can do is to define problem and simulations very explicitly and leave it to the reader to judge if justice has been done to both algorithms, which have been specified in sections 2 and 4, respectively.

If a square lattice is mapped onto a continuous input square (a typical problem for the SOM without features), the correct solution is not obvious. For a fair comparison, however, the quality of a solution must be assessed by objective means. The solution is not unique, because mirror reflections and rotations by multiples of 90° in one of the layers have, of course, the same quality.

To avoid these problems we have chosen the above-mentioned mirror-problem as a benchmark. The setup consists of two square layers X and Y of $N \times N$ neurons that in addition carry features $f \in \{1, \ldots, F\}$. The feature distribution in X is chosen at random, but symmetric[1] patterns are discarded. The distribution in Y is identical to the one in X except for a mirror reflection at the horizontal or vertical symmetry axis of the square or a rotation by multiples of 90° around the center. Because of the lack of symmetry, the feature distributions induce a unique neighborhood-preserving mapping from X to Y. The benchmark task for the self-organizing algorithms is to find this mapping given only the feature distributions. *Similarity* of features of neurons $x \in X$ and $y \in Y$ is defined as all-or-nothing for this benchmark (for practical applications smooth similarity functions are usually more suitable):

$$T(\vec{x}, \vec{y}) = \delta\left(f(\vec{x}), f(\vec{y})\right) = \begin{cases} 1 & \text{if } f(\vec{y}) = f(\vec{x}) \\ 0 & \text{otherwise}\,. \end{cases} \tag{1}$$

Due to the discrete lattices in both layers and the symmetry-breaking features attached to the neurons, neighborhood preservation is clearly defined and the optimal solution is known beforehand, which gives a

[1]This includes symmetries that only arise if the layer topology is a torus, because DLM actually uses that topology. For a discussion of the treatment of boundaries see the end of section 5

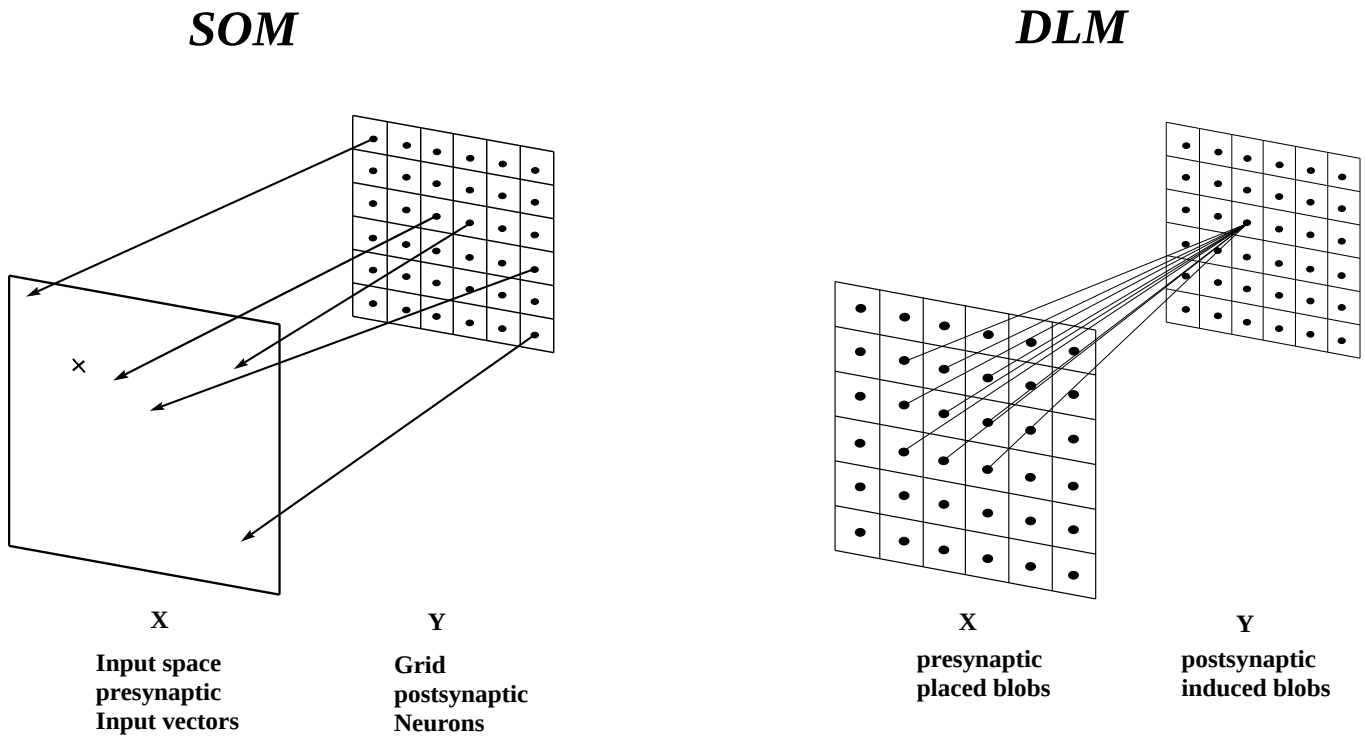

Fig. 1: The setup for the benchmark problem. A correct mapping must be found between two layers, both of which carry features (which are not shown here). A Self-organizing feature map (left hand side) and dynamic link matching (right hand side) are both applied to the problem, where the learning rates include the feature similarity between both ends of a link.

straightforward error measure that can be monitored through the whole process. Let $\mathcal{M}_t(\vec{x})$ be the position where neuron $\vec{x}$ points in layer Y at time t, and $\mathcal{M}_{opt}(\vec{x})$ be the optimal mapping. Then the error at time t will be

$$E(t) = \sum_{\vec{x}} \left(\mathcal{M}_t(\vec{x}) - \mathcal{M}_{opt}(\vec{x})\right)^2 . \tag{2}$$

For the DLM, the vector $\mathcal{M}_t(\vec{x})$ is defined as the center of mass for all links $J(\vec{x}, \vec{y})$ emerging from neuron $\vec{x}$ (see (6)).

Note that in this benchmark problem the conditions **M1** through **M3** are not conflicting, and the solution is unique. For real data, where they might be conflicting, the optimal solution is defined only by the behavior of the algorithms themselves. This may not be very satisfactory, but the analytical treatment of either algorithm still leaves much to be desired (Kohonen, 1997).

For the $N \times N$-size benchmark problem a particular solution is said to have *converged* if the average position error $E(t)/N^2$ is below a threshold ε. The number of features F is a useful parameter to control the difficulty of the problem. For $F = 1$ it is impossible to find non-symmetric distributions. The more different features there are the fewer ambiguities are encountered, and the easier the correspondence problem becomes. For large F the correct solution can be found by just searching the identical feature in the other layer. For all simulations, we have used $F = 10$ equally distributed features.

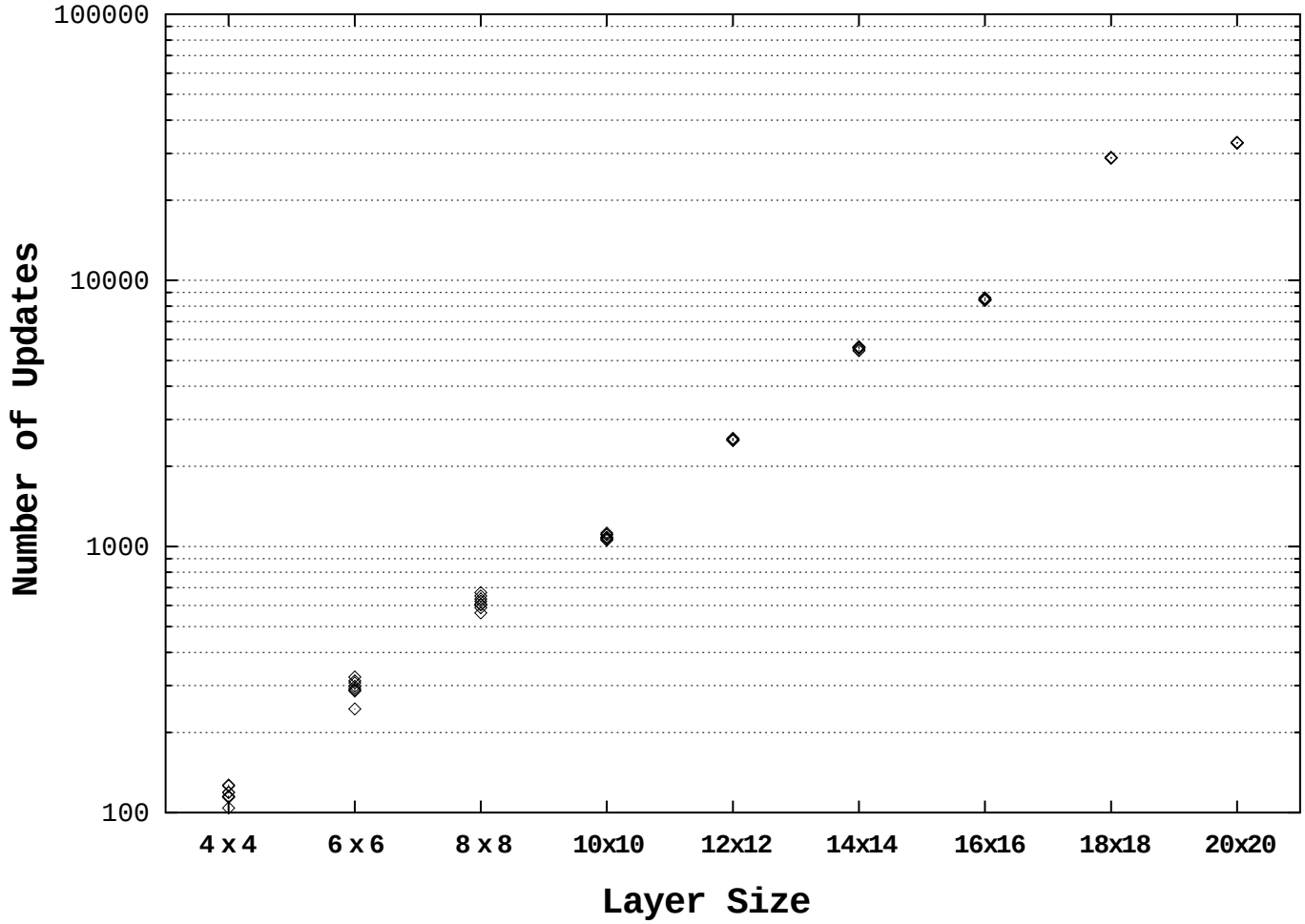

Fig. 2: Scaling behavior of the SOM algorithm on the mirror benchmark. Only runs with the optimal parameter set for each problem size are shown. The vertical spread of data points is due to different random seeds. A straight line fits the data well, hinting at exponential scaling with problem size.

3 A modified Self Organizing Feature Map (SOM)

In order to apply the SOM to the correspondence problem we have identified the discrete neuron layer with a discrete layer of neurons X and the input space with another discrete layer Y. Furthermore, we have included the feature similarity $T(\vec{x}, \vec{y})$ into the learning rule of an otherwise unmodified SOM-algorithm (Kohonen, 1982; Kohonen, 1990). The algorithm is as follows.

Initialize pointers from Y to X randomly (one pointer for each element of Y) and iterate the following until convergence:

1. Choose an arbitrary element $\vec{x}_0$ of X.
2. Select the element $\vec{y}_0$ of Y such that $\mathcal{M}(\vec{y}_0) - \vec{x}_0$ is minimal.
3. update $\mathcal{M}$ by the following increment

$$\Delta\mathcal{M}(\vec{y}) = \lambda \exp\left(-|\vec{y} - \vec{y}_0|^2/2\sigma^2\right) (\vec{x}_0 - \mathcal{M}(\vec{y}_0))\, T(\vec{x}_0, \vec{y}_0))\,. \tag{3}$$

In the benchmark, T can only take the binary values 0 and 1, so that iteration steps for neurons with unequal features are without effect. Therefore, we have optimized the algorithm by skipping all such iteration steps and choosing $\vec{y}_0$ directly among the neurons with the same features as $\vec{x}_0$. Only those "effective" iteration steps will be counted in our results.

Practically all applications of SOM require dynamic modification of at least one of the parameters λ and σ to assure convergence. Different decreasing schemes (linear, exponential, $1/t$) are in use, but are known (Kohonen, 1990; 1997 p. 88) to yield the same general behavior of the algorithm. According to our own

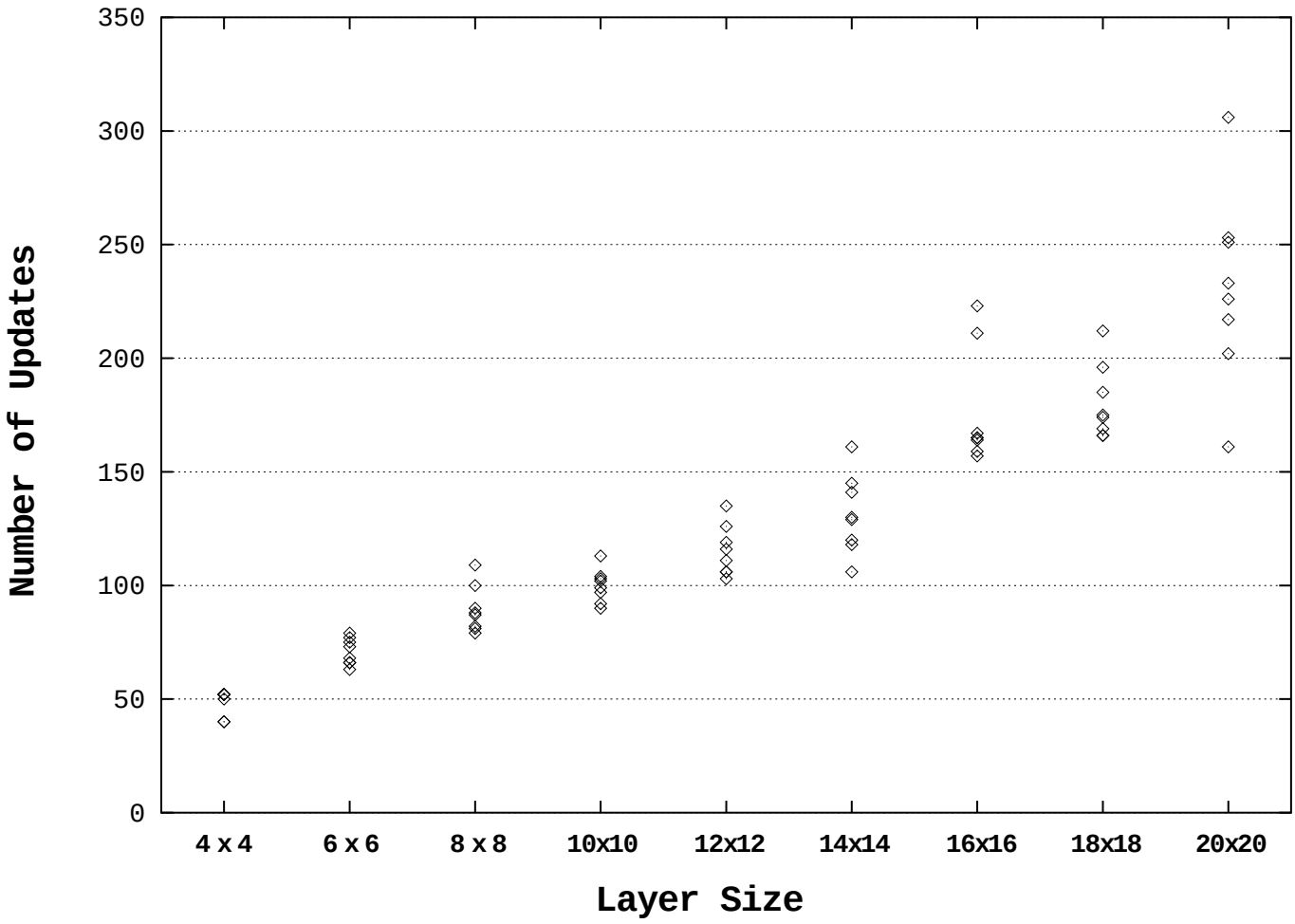

Fig. 3: Scaling behavior of the DLM algorithm on the mirror benchmark. The vertical spread of data points is due to different random seeds. A straight line is compatible with the data, this time suggesting linear scaling with problem size.

experience the actual time course of the parameters matters less than a combination of a good starting value and the coincidence of the map reaching its stable state and the parameters reaching zero. The necessity for parameter adaptation during development introduces new parameters describing the decreasing speed. It is clear that neither learning rate nor neighborhood width may drop below zero. For a linear decreasing scheme, which we have used throughout, a complete parameter set describing the algorithm consists of $\lambda_0,\ \Delta\lambda, \sigma_0, \Delta\sigma$.

Extensive experiments (Behrmann, 1993) have shown that the performance could not be improved by decreasing λ, and we have kept it constant at $\lambda = 1$, but the width parameter σ_0 and its decrease rate $\Delta\sigma$ had to be chosen carefully. Both were optimized individually for each problem size ($N = 4, \ldots, 20$) by scanning a reasonable parameter range (of 40 parameter pairs) with 10 executions of SOM each. From the best parameter setting we show in Fig. 2 only the 8 best results among the 10 runs. The reason why we do not show all 10 results is that even with this careful parameter selection the SOM did not converge in all cases. This implies that the parameters work well but are not yet on the safe side for convergence and longer convergence times must be expected if that is required. The results are consistent with an exponential scaling of the number of update steps with the problem size (see Fig. 2). The complete parameter set was as follows: The learning rate was $\lambda = 1.0$, the initial width of the Gaussian defining the neighborhood was varied $\sigma_0 \in \{0.1\,N, 0.2\,N, \ldots 1.0\,N\}$, the decrease $\Delta\sigma$ varied between $4 \cdot 10^{-7}\,N$ and $10^{-4}\,N$. In order to avoid underflow problems the decrease of σ was cut off at $\sigma_{min} = 10^{-6}\,N$. The number of features was kept at $F = 10$, the layer size was varied $N \in \{4, 6, 8 \ldots, 20\}$. The error threshold for convergence was $\varepsilon = 1/640$ throughout.

4 Dynamic Link Matching (DLM)

In the DLM scheme layer X is fully connected with Y by a matrix $J(\vec{x}, \vec{y})$ of dynamical links. Their development is governed by a Hebbian rule with competition and influence of feature similarity. In other

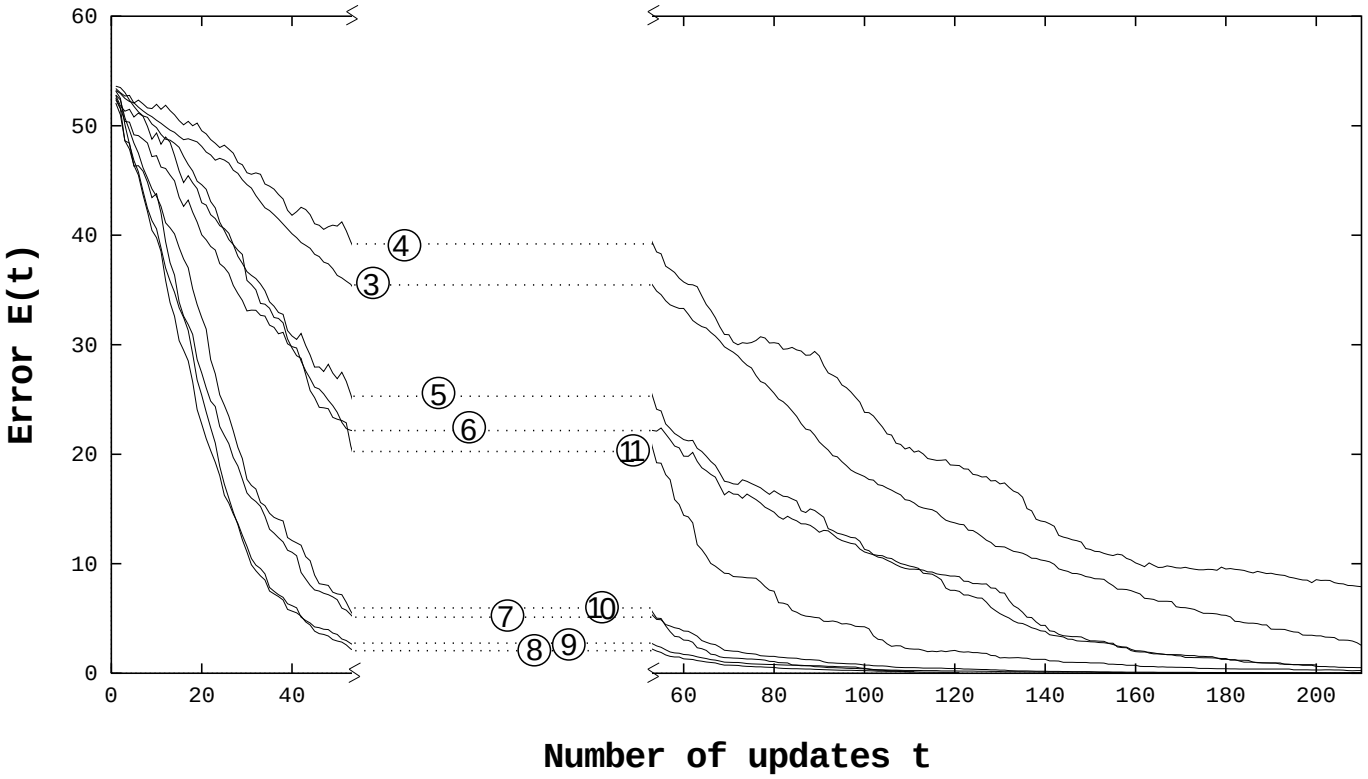

Fig. 4: Dependence of the convergence speed of DLM on the blob size. The error is plotted against time for a 12×12-layer for different blob sizes $B \times B$ (square blobs) (B is the number in the circle). Best results are obtained for $B = 8$ and $B = 9$ which is closest to blobs covering half of the layer area.

words, links between pairs of neurons which have *similar features* and are *active at the same time* will be strengthened, others decay. Neighborhood preservation in DLM is achieved by ensuring that in each layer only *one connected subregion* of a given form and size, which we will call a *blob*, can be active at a time. This is a way to code neighborhood in the layer as common activity in the same time slot. A blob in layer X excites layer Y by means of the dynamic links $J(\vec{x},\vec{y})$. Layer Y supports only blobs of the same form and size as X. The links only influence the *position* of the blob in Y. This position can be calculated analytically (Konen et al., 1994). Now only the neurons in the blob in X and the resulting blob in Y are active and can strengthen their links in the following update step according to their feature similarities. An activity blob is a unimodal nonnegative function $b(\cdot)$ of the layer neurons. In our simulations we have chosen it to be 1 inside a square of size $B \times B$ and 0 outside. Then, the concrete algorithm runs as follows ($\ominus$ stands for componentwise subtraction modulo N:

1. All links are initialized to $1/N^2$.
2. A position $\vec{x}_0 \in X$ is chosen at random, a blob is placed there, and the resulting blob position $\vec{y}_0 \in Y$ is calculated such as to minimize the *potential*
$$V(\vec{y}_0) = -\sum_{\vec{y}} \sum_{\vec{x}} J(\vec{x},\vec{y})\, T(\vec{x},\vec{y})\, b(\vec{x} \ominus \vec{x}_0)\, b(\vec{y} \ominus \vec{y}_0)\,. \tag{4}$$
3. The activities in X and Y are now blobs positioned at $\vec{x}_0$ and $\vec{y}_0$, respectively, and the links $J(\vec{x},\vec{y})$ are updated by the learning rule:
$$\Delta J(\vec{x},\vec{y}) := \lambda\, J(\vec{x},\vec{y})\, T(\vec{x},\vec{y})\, b(\vec{x} \ominus \vec{x}_0)\, b(\vec{y} \ominus \vec{y}_0)\,. \tag{5}$$

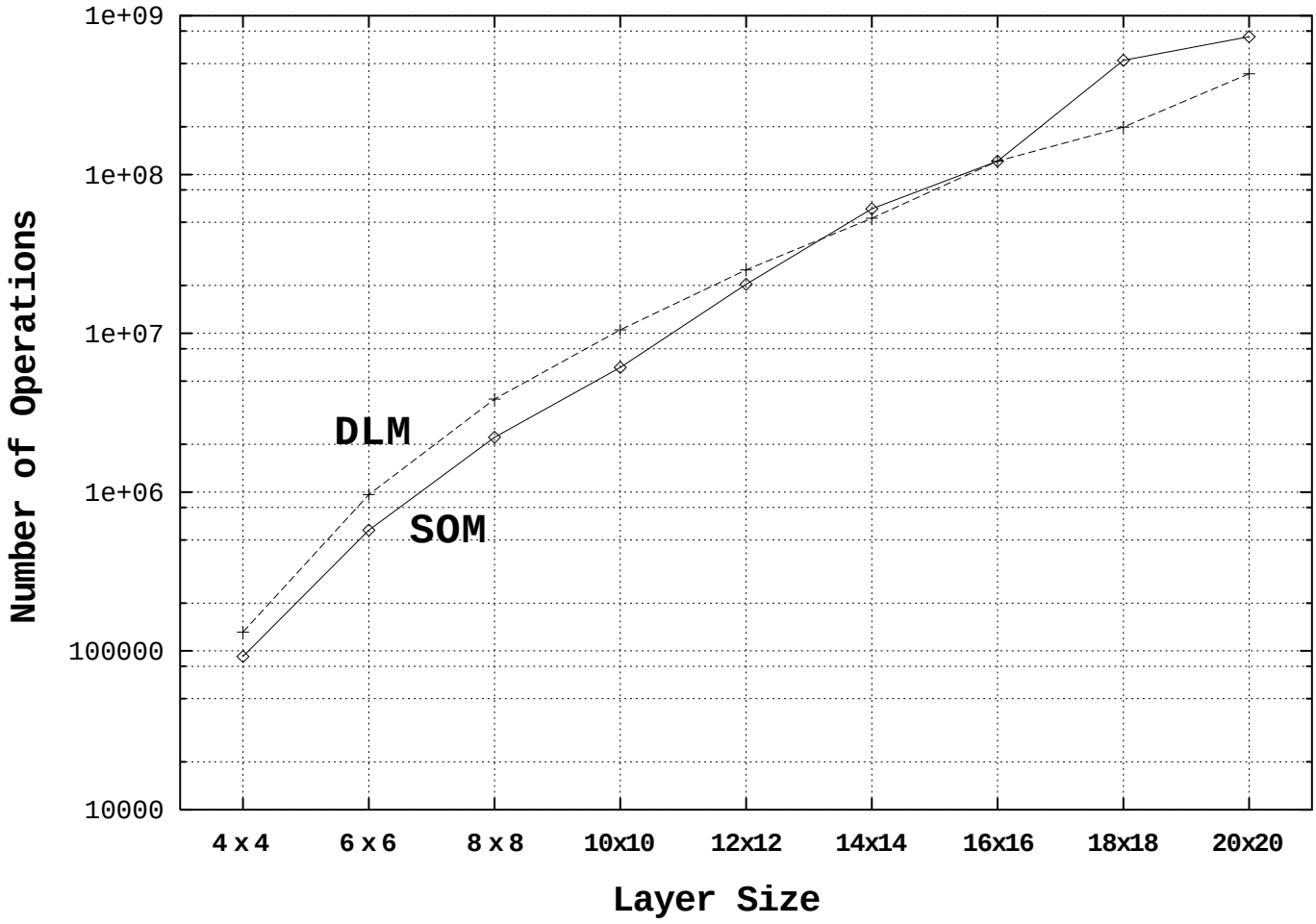

Fig. 5: Floating point operations required for the SOM and DLM on the mirror benchmark. The data indicate that DLM is faster for layers larger than 16×16.

4. The updated links are first normalized by division by $\sum_{\vec{x}} J(\vec{x},\vec{y})$ and then by division by $\sum_{\vec{y}} J(\vec{x},\vec{y})$.

Steps 2 through 4 are iterated until convergence.

The algorithm converges to a state where each neuron in X has *one* strong link to a neuron in Y and all the other links are very small. In the early stages of DLM the mapping is not unique. The error function (2), however, requires a unique mapping. For that reason, the *center of mass* for all links $J(\vec{x},\vec{y})$ emerging from neuron $\vec{x}$ has been used:

$$\mathcal{M}_t(\vec{x}) = \frac{\sum_{\vec{y}} \vec{y} J(\vec{x},\vec{y})}{\sum_{\vec{y}} J(\vec{x},\vec{y})}\,. \tag{6}$$

The simulations of DLM on the mirror benchmark have shown that its parameters are fairly simple to adjust. There is no necessity to decrease the learning rate λ or change the blob size B during iterations in order to obtain convergence. The form of the blob (circle, square, ...) does not influence convergence. Having chosen the learning rate λ as large as possible without impeding convergence, the only relevant parameter is the *blob size* B. Simulations in (Behrmann, 1993) have shown that convergence is fastest if a blob covers half the layer area. We see also from Fig. 2 that for a 12x12 layer square blobs with a side length of 8 or 9 units give fastest convergence. (A square blob of length $\sqrt{72} = 8.49$ would cover exactly half the layer area.) The blob size $B = N/\sqrt{2}$, which we used for all DLM simulations, can be shown to maximize the average information gain per iteration step (see appendix for details).

The experimental conditions have been identical to those for the SOM. The scaling behavior, however, was different. On variation of the problem size N the number of iteration steps needed for convergence increase only linearly with N (Fig. 2), in sharp contrast with the exponential behavior of the SOM (Fig. 2). The complete list of parameters used for the DLM simulations is: $B \approx N/\sqrt{2}$ for the blobsize, $\lambda = 1.0$ for the learning rate, a constant feature number $F = 10$, a varying layer size $N \in \{4, 6, 8 \ldots, 20\}$, and a constant error threshold $\varepsilon = 1/640$.

5 Conclusions

During our experiments we have encountered fewer difficulties in adjusting the parameters for DLM than for the SOM. Nevertheless, we have invested considerable effort to tailor both algorithms for the benchmark problem. For DLM the essential parameter is the blobsize B which should be such that the blobs cover half the layer (see section 5 for a plausibility argument in favor of this choice). For SOM, the parameter choice had to be done much more carefully, and even after that some runs without convergence remained. That shows that the parameters could hardly be modified for faster convergence without losing convergence proper.

Figs. 2 and 2 show the number of iterations required to solve the mirror problem. The comparison of these two figures is not completely fair, because the single update steps for DLM are much more complicated ($O(N^4)$) than the ones for the SOM ($O(N^2)$). In order to show the actual execution time for concrete layer sizes we have plotted the number of floating point operations required to reach convergence in Fig. 2. This figure indicates that DLM converges faster once the layer size exceeds 16×16. We conclude that the convergence time for DLM will scale as N^4 and the one for the SOM as $\exp(N)$. For an estimation of the whole effort, the figures for SOM ought to be multiplied by a factor of 40, the number of parameter combinations we tested. The price paid for the faster convergence of DLM is of course the memory requirement of $O(N^4)$ as compared with $O(N^2)$. For a "mixed" complexity measure containing terms for processing time and memory requirement this would still not change the exponential vs. polynomial behavior.

A question that cannot be ignored in the comparison of neuronal algorithms as models for perception is the time required on parallel machines. Given arbitrarily many parallel processors the single update steps can be executed in constant time for both algorithms. The update steps themselves cannot be parallelized completely. To what extent partial parallelization (epoch learning) can be applied is unclear at this point. We therefore expect that on massively parallel machines DLM will scale linearly with the layer size, whereas the SOM will retain its exponential behavior. This is not particularly grave for models of brain development, where long convergence times are acceptable. For fast processes during perception, our results show that

SOM is not suitable. If DLM with its fast convergence is applicable to other domains of SOM such as finding low-dimensional submanifolds in high-dimensional feature spaces must be left to future research.

A possible objection to the comparison we have presented is that the topologies of the neuronal layers are different. SOM is always executed with the normal topology of the square, DLM with torus topology. The reason for this is purely practical. SOM with torus topology very often does not converge with the parameters used for the square topology. It appears that its self-organization is supported by the influence of the borders. DLM on the other hand does not show good self-organization *with* boundaries, mainly because the neurons there have a lower chance of being hit by a blob, and also the derivation of the position of the induced blob which is used in the placement of the Y-blob (step 2 of the DLM-algorithm) is no longer valid (see (Konen et al., 1994) for the derivation.)

Particularly annoying in this context is the fact that the definition (6) of the center of mass is not independent of the choice of coordinate system on the torus. It is indeed ill-defined in the beginning, when all links are about equal in size. However, it has the crucial property that for a fully converged mapping the resulting pointers are identical to the actual links. We consider the early steps, when it might indeed give strange results, as not important for the convergence speed. Later on, only a few neighbors of each link will have significant size, so the problem occurs only at the boundary. If a link there should be dragged towards the center of the square, although the proper position ought to be near the boundary, this is simply a distortion from the ideal mapping and can, consequently, only result in a disadvantage for DLM. Furthermore, the very fact that in our experiments DLM *does* converge using that measure, shows that this effect is rather negligible.

To summarize: We have tested SOM and DLM on a simple two-dimensional matching problem, the correspondence problem for the case of identical but mirrored or rotated patterns. We encountered difficulties in the parameter adjustment of SOM in the sense that a considerable parameter range had to be checked by trial and error. In the case of DLM, all parameters could be fixed by a simple rule and led to satisfactory results. After scrupulous parameter adjustment for each layer size, measurements of convergence times for SOM showed an exponential dependence on the layer size, while the results for DLM were consistent with a polynomial one without any need for parameter adjustment by trial and error. It appears safe to conclude that for matching large patterns without reduction of dimension DLM should be preferred over SOM. We can not be absolutely sure that the optimal parameter sets for the SOM have actually been tested, especially a different choice of decrease schedule for the neighborhood width could possibly bring some improvement. Consequently, we expect our results to stimulate further research on the important question of how optimal parameters for SOM can be found. Only then can the question of the scaling of convergence time with problem size be tackled more seriously than by presenting examples.

AppendixA The optimal blob size for DLM

It is intuitively clear that there must be an optimal blob size which leads to fastest convergence of the DLM: Too small blobs let only very few links learn in each iteration while a blob as large as the whole layer allows only learning according to the feature similarities.

With a simple argument we will demonstrate why the size of a blob which covers half of the area of the neuron layer is optimal. N^2 is the number of neurons in the whole neuron layer and B^2 be the number of neurons in each blob. Let us assume that we have in a certain iteration step a pair of blobs in X- and Y-layer which are correctly positioned in the sense of the underlying mapping (this is usually the case after a short initial phase).

What can be learned in this constellation about the correct mapping $\mathcal{M}_{opt}$? At least we learn something about neurons $\vec{x} \in X$ and $\vec{y} \in Y$ which are definitely *not* connected, namely the ones where one partner is inside the blob and the other one outside the other blob. Without a-priori knowledge about the link matrix, then we learn from the blob pair that all links that connect cells inside a blob with cells outside a blob, i.e. $2B^2(N^2 - B^2)$ links can *not* belong to the correct mapping.

Now, the most information from such a blob pair is gained if this is maximal. Setting the derivative to zero yields $B^2 = 2N^2$, thus the blob area must cover half of the layer.

This argument assumes that the time when the blobs are still at non-corresponding positions does not influence the convergence time. From a lot of simulations we have seen that it is very frequently zero (blobs are at corresponding positions from the very first step). It is also unclear how the overlaps of blobs between different iterations should be accounted for. The simulation results (see Fig. 2 and (Behrmann, 1993) for full details) indicate that both effects are probably negligible for practical purposes.

From this interpretation it can be conjectured that an antihebbian learning rule could give better results. We will certainly try this suggestion by one of the referees for future versions of DLM.

The blobs of DLM can also be interpreted in the following way. The goal is to relax the continuity constraint **M2**, because continuity between discrete spaces is trivial. A topology (and thus continuity) is usually defined in terms of the collection of open sets. This can hardly be used to get a workable definition of neighborhood preservation, because again, all sets are open in discrete spaces.

However, topology can also be defined in terms of neighborhood systems. This leads to Cauchy's definition of continuity: *A function f is continuous if and only if for each neighborhood V of a point $f(x)$ there is a neighborhood U of x such that $f(U) \subseteq V$.* DLM can be interpreted as trying pairs of neighborhoods and making sure that the points inside the neighborhood in X are hindered from being mapped outside the neighborhood in Y. Neighborhoods smaller than the blobs are created virtually by the intersection of two or more blobs from the succession of blobs during the self-organizing process.

References

Behrmann, K.-O. (1993). Leistungsuntersuchungen des "Dynamischen Link-Matchings" und Vergleich mit dem Kohonen-Algorithmus. Technical Report IR-INI 93-05, Ruhr-Universität Bochum, Master's thesis, Universität Karlsruhe.

Bellando, J. and Kothari, R. (1996). On image correspondence using topology preserving mappings. In *Proceedings of the International Conference on Neural Networks, Washington, 1996*, pages 1784–1789.

Goodhill, G. J., Finch, S., and Sejnowski, T. J. (1995). Quantifying neighborhood preservation in topographic mappings. Technical Report INC-9505, Institute for Neural Computation, UCSD, La Jolla, CA 92037, USA.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Kohonen, T. (1990). The self-organizing map. *Proc. IEEE*, 78:1464–1480.

Kohonen, T. (1997). *Self-Organizing Maps*. Springer Verlag, Berlin, Heidelberg, New York, 2nd edition.

Konen, W., Maurer, T., and von der Malsburg, C. (1994). A fast dynamic link matching algorithm for invariant pattern recognition. *Neural Networks*, 7(6/7):1019–1030.

Konen, W. and von der Malsburg, C. (1993). Learning to generalize from single examples in the dynamic link architecture. *Neural Computation*, 5:719–735.

Lades, M., Vorbrüggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Würtz, R. P., and Konen, W. (1993). Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311.

Sejnowski, T., Kienker, P., and Hinton, G. (1986). Learning symmetry groups with hidden units: Beyond the perceptron. *Physica D*, 22:260–275.

Willshaw, D. J. and von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society, London*, B 194:431–445.

Wiskott, L. (1996). *Labeled Graphs and Dynamic Link Matching for Face Recognition and Scene Analysis*. Reihe Physik. Verlag Harri Deutsch, Thun, Frankfurt am Main.

Wiskott, L. and von der Malsburg, C. (1996). Face recognition by dynamic link matching. In Sirosh, J., Miikkulainen, R., and Choe, Y., editors, *Lateral Interactions in the Cortex: Structure and Function*. The UTCS Neural Networks Research Group, Austin, TX, Electronic book, ISBN 0-9647060-0-8, http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96.

Würtz, R. P. (1995). *Multilayer Dynamic Link Networks for Establishing Image Point Correspondences and Visual Object Recognition*. Verlag Harri Deutsch, Thun, Frankfurt am Main.

Würtz, R. P. (1997). Object recognition robust under translations, deformations and changes in background. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):769–775.