GRAPH RECOGNITION BY NEURAL NETWORKS

G. DREYFUS
Ecole Supérieure de Physique et de Chimie Industrielles
de la Ville de Paris
Laboratoire d'Electronique
10 rue Vauquelin
F - 75005 PARIS
FRANCE

and

A. ZIPPELIUS
Institut für Theoretische Physik,
Georg-August Universität
Bunsenstrasse 9
D - 3400 GÖETTINGEN
FRG

ABSTRACT

The present paper addresses the problem of graph recognition; since graph representations are central in Artificial Intelligence, this problem has attracted considerable interest. In contradistinction to the widespread opinion that neural networks are unable to handle structured data, we show that graph recognition can indeed be performed by making use of a variant of a neural network model suggested recently. Simulations were performed to investigate the capabilities of the network with respect to system size, to the number of stored graphs and to the noise level of the input.

1. INTRODUCTION

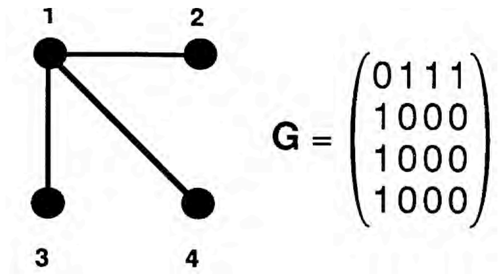
A central problem in Artificial Intelligence as well as in the field of neural networks is that of data representation: a proper representation is, in most cases, the key to the solution of a given problem. Up to now, the representations used by neural networks were essentially low-level typically, in problems of visual pattern recognition, one neuron would code for one picture element. Conversely, symbolic computing uses high-level, relational representations of knowledge; in pattern recognition or scene analysis, structural descriptions of the pictures are used, such as graphs describing the relations between objects in a scene, between edges of an object, etc [1]. It has often been argued that neural networks, being unable to handle graphs, are restricted in scope to handling unstructured representations. A first attempt at a relational treatment of pictures was suggested by C. von der Malsburg and E. Bienenstock (see in the present proceedings the paper by E. Bienenstock and R. Doursat, and references therein); it is based on the conjecture of fast synaptic plasticity, and is aimed at a relational treatment in low-level vision, i.e. on raw images.

An alternative approach was taken in Ref. 2; a neural network model was suggested for the recognition of isomorphisms of unlabelled graphs, which is known to be an NP-complete problem in its general form. In the present paper, it is shown that a variant of the latter model has the ability of performing error-correcting isomorphism. We will first outline the original method, and subsequently describe the variant that is suggested here; simulation results will be presented, and the virtues and limitations of the method in its present state will be assessed. Further developments and improvements will be outlined.

2. DEFINITIONS AND MODELS

2.1 General description:

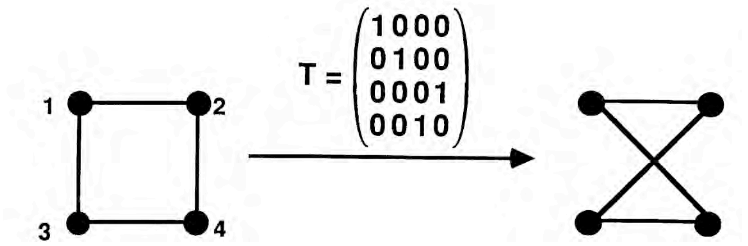
In all the following, we consider graphs with a maximum of N nodes, labeled in an arbitrary way from i = 1 to N; thus, there are N(N-1)/2 possible undirected edges. The presence or the absence of an edge between nodes I and j in a given graph is denoted by a variable G1 which takes on the value +1 if edge (ij) is present in the graph and O if it is absent. A graph is therefore represented by its connectivity matrix G, as shown in Fig. 1.



$$G = \begin{pmatrix} 0111 \\ 1000 \\ 1000 \\ 1000 \end{pmatrix}$$

Figure 1

Two graphs are isomorphic if there exists a permutation of the nodes that makes both graphs identical (Figure 2).



$$T = \begin{pmatrix} 1000 \\ 0100 \\ 0001 \\ 0010 \end{pmatrix}$$

Figure 2
Two isomorphic graphs: permutation (1, 2, 4, 3) maps the left graph onto the right one.

The problem that is addressed here is the following *given a collection of prototype graphs, find whether an unknown graph is isomorphic to one of the prototype graphs; in the negative, find whether the unknown graph is topologically close to one of the prototype graphs.*
This problem can also be expressed as: is it possible to find a permutation of the nodes of the input graph which maps the latter onto one of the prototype graphs, or which makes it as similar as possible to one of the prototype graphs?

This can be performed by the simultaneous operation of two coupled neural networks:

- a Hopfield -Tank optimizer [3] to find a graph that is obtained by a permutation of the nodes of the input graph,
- a Hopfield associative memory [4], which determines whether the graph found by the first net, is identical with or close to one of the prototype graphs.

The operation of the first network (termed "preprocessor") can be understood as follows: a Hopfield -Tank optimizer with N^2 neurons can be designed so as to find a permutation of N nodes which minimizes the distance between a given graph and a graph produced by the permutation of the nodes of the input graph. In other words, the network, left to evolve spontaneously, would reach a state of minimum energy that codes for a permutation of the nodes of the input graph mapping the latter onto a given graph. This operation is very similar to that used to solve the traveling salesman problem, where the neural network is able to find a permutation of N towns which minimizes the length of a tour.

The second network (termed "memory" network) is a standard associative net with N(N-1)/2 neurons; its synaptic couplings J_{ij} are computed in such a way that the prototype graphs are global energy minima of the network [5]. Once initialized in a state (coding for a graph), it will evolve until it reaches a state of minimal energy; therefore, it has the ability to associate the initial graph to one of the prototype graphs.

When both networks evolve simultaneously, they will both be driven to a state of minimal energy, thereby,
i - finding, by operation of the preprocessor network, a permutation of the nodes of the input graph,
ii - retrieving, by operation of the memory network, the prototype graph which is topologically closest to the graph generated by application of the above permutation to the unknown graph.

2.2 Mathematical analysis

As stated above, the system consists of two coupled networks (the preprocessor and the memory). In addition, a layer of input units (the receptor) is used for defining the input graph; it has N(N-1)/2 units, each of them coding for the presence or the absence, in the input graph, of an edge between two nodes. The nodes are numbered from 1 to N, and the state of the unit coding for edge (ij) is denoted by G_{ij}, which can take on the values 1 or 0, as described in the previous section.

The memory network is designed to store and retrieve graphs which have the same characteristics as those defined on the receptor layer: the N nodes are labeled from α = 1 to N; the state of neuron $s_{\alpha\beta}$ codes for the presence ($s_{\alpha\beta}$ = 1) or the absence ($s_{\alpha\beta}$ = 0) of an edge between nodes α and β. The N(N-1)/2 neurons in the memory interact through the Hamiltonian

$$H_M(s) = -\sum_{\alpha,\beta,\gamma,\delta} \left(2s_{\alpha\beta} - 1\right) J_{\alpha\beta\gamma\delta} \left(2s_{\gamma\delta} - 1\right)$$

The synaptic couplings $J_{\alpha\beta\gamma\delta}$ are determined by a suitable learning algorithm from the graphs to be learnt $\left\{\xi^{\nu}_{\alpha\beta}\right\}$.

In order to recognize isomorphic graphs, we have to compare two graphs, one of which - the input graph G - is defined on the receptor, and the other one is encoded as the state $\{s_{\alpha\beta}\}$ of the memory. In particular, we want to know whether there exists a transformation T which maps all nodes of the graph described by $\{G_{ij}\}$ on the nodes of the graph defined by $\{s_{\alpha\beta}\}$. These transformations consist of all the different labelings of the N points of the receptor: each of them can be represented by a (N, N) matrix $t_{i\alpha} \in (0,1)$. Since one node i is mapped exactly onto one node α, there is one and only one element equal to 1 in each row and in each column of T (see Figure 2). In order to compare the topology of two graphs, one can use the minimum of their Euclidean distance

$$\begin{aligned} D^2 &= \min_T d^2\left(s, T^+GT\right) \\ &= \min_T \frac{1}{2}\sum_{\alpha,\beta}\left(s_{\alpha\beta} - \sum_{i,j} t_{i\alpha} t_{j\beta} G_{ij}\right)^2 \end{aligned}$$

Two graphs are isomorphic if D = 0.

In the present approach, the transformations are represented as states of neural activity: the elements $t_{i\alpha}$ of the transformation matrix are encoded into the states of the N^2 neurons of the preprocessor. The spontaneous evolution of the preprocessor leads to a state that is a minimum of the function

$$H_P = d^2\left(s, T^+GT\right) + H_{penalty}$$

The last term is introduced to enforce the constraints

$$\sum_{\alpha} t_{i\alpha} = 1 \text{ and } \sum_{i} t_{i\alpha} = 1$$

Additional details are given in Ref. 2.

Recognition of isomorphic graphs is achieved by the combined action of the preprocessor and the memory. During the learning phase, a set of prototype graphs $\left\{G^{\nu}_{ij}\right\}$ is defined on the receptor; each of them is transformed into a state $\xi^{\nu}_{\alpha\beta}$ of the memory network by:

$$\xi^{\nu}_{\alpha\beta} = \sum_{i,j} t^{\nu}_{i\alpha} t^{\nu}_{j\beta} \left(2G^{\nu}_{ij} - 1\right)$$

where each transformation T^{ν} is arbitrary; the simplest choice is the identity transformation, so that the graphs stored in the memory are identical to the prototype graphs; the choice of the T^{ν}'s will be discussed below. The projection rule [5] is used to store the patterns; if the latter are linearly independent, it can be expressed as

$$J_{\alpha\beta\gamma\delta} = \frac{2}{N(N-1)} \sum_{\mu,\nu} \xi^{\nu}_{\alpha\beta} C^{-1}_{\mu\nu} \xi^{\nu}_{\gamma\delta}$$

where

$$C_{\mu\nu} = \frac{2}{N(N-1)} \sum_{\alpha,\beta} \xi^{\mu}_{\alpha\beta} \xi^{\nu}_{\alpha\beta}$$

and $J_{\alpha\beta\alpha\beta} = 0$.

After the network has learnt p patterns, it is presented with a graph G that is an isomorphism of one of the stored graphs. In order to recognize the isomorphic prototype, the neurons of the preprocessor and those of the memory are allowed to evolve simultaneously, thereby generating isomorphisms of the input graph and retrieving one of the learnt patterns. The evolution of the states $\{t_{i\alpha}\}$ and $\{s_{\alpha\beta}\}$ is determined by the total Hamiltonian:

$$H = \lambda d^2\left(s, T^+GT\right) + H_M(s) .$$

The learnt graphs are the global minima of H_M, and the distance d vanishes if the preprocessor has found a transformation T that maps all nodes of the input graph G onto the nodes of the graph represented by the state of the memory s. Thus, the global minima of H are given by

$$\sum_{i,j} t_{i\alpha} t_{j\beta} G_{ij} = s_{\alpha\beta} = \frac{1}{2}\left(\xi^{\nu}_{\alpha\beta} + 1\right)$$

Therefore, an isomorphism is recognized if a global minimum is found.

2.3 Operational model

In this section, we introduce a variant of the above model, which we used for the simulations. The motivation for this variant is twofold:

a) one problem with the above approach lies in the use of a Hopfield-Tank optimizer to find a permutation of the nodes of the unknown graph. It is well known that such networks would be potentially very powerful, by virtue of their massive parallelism [6], if they were actually manufactured in silicon; however, there is only one known realization of such a network, and it is unlikely that it will become widely available. Therefore, one has to perform computer simulations that are extremely slow and make the whole thing impractical for networks of realistic size. In order to avoid this, we consider a limiting case of the model: we take $\lambda >> 1$, so that states of the memory network which are not isomorphisms of the input graph have a very high cost, hence are very unlikely to occur; in the limiting case $\lambda \rightarrow \infty$, the only possible states for the memory network code for isomorphisms of the input graph, so that the energy which is minimized by the system is given by:

$$H = -\sum_{\alpha,\beta,\gamma,\delta} \sigma_{\alpha\beta} J_{\alpha\beta\gamma\delta} \sigma_{\gamma\delta}$$

with:

$$\sigma_{\alpha\beta} = \sum_{i,j} t_{i\alpha} t_{j\beta} \left(2G_{ij} - 1\right)$$

b) In addition to recognizing isomorphisms of the prototype graphs, we want the network to perform error-correcting isomorphism: if the input graph differs from an isomorphism of a prototype by some amount of random noise, we want the system to be able to recognize it.

In order to satisfy the above two requirements, we suggest a modified procedure which consists of two steps:

1 - in a first step, a search is performed for a graph, isomorphic to the unknown one, which minimizes the energy of the network. This minimization procedure should lead to a global minimum, which corresponds to the isomorphism which minimizes the distance between the transformed input graph and one of the prototype graphs; if the final energy is that of a prototype graph, the procedure stops;

2 - if no prototype graph is found, the neural network is initialized in the state coding for the graph found in the previous step, and left to evolve under the usual neural dynamics, without restrictions on the state of the neurons: if the state found by the previous search is not very different from one of the prototype graphs, the network will reach one of the global minima, which will be the prototype which is topologically closest to the unknown graph.

Thus, the two steps are distinguished by the dynamics of the network: in the first step, the states are restricted to isomorphisms of the input graph, whereas, in the second state, no such restriction exists.

3. SIMULATIONS

Step 1 can be carried out by any suitable iterative search procedure; the simulations presented here use simulated annealing [7], with parameter settings as in Ref. 8. The elementary move is the exchange of two nodes of the graph. Step 2 uses a neural network with sequential dynamics.

The prototype states used in the following experiments are shown on Figure 3. They have a maximum of 9 nodes, and represent a variety of topologies for planar and non-planar graphs. It is known that the error-correcting abilities of a neural network are improved if the Hamming distance between the stored patterns is large. In order to increase the distance between the stored patterns that code for the graphs, we first transformed the prototypes by applying to the nodes of each prototype ξ^{ν} a random permutation T^{ν}. A random permutation is a simple choice, presumably not an optimal one. Note, however, that the basins of attraction of the whole system are not governed by Hamming distance alone (see the following discussion).

3.1 Recognition of isomorphic graphs
In a typical set of experiments, fifty unknown graphs, all of them isomorphic to graph G_1, were generated. All fifty graphs were presented as inputs to networks whose neurons encode the links of a lattice of N points, and in which we stored p prototypes the following combinations were investigated: N=9, p=6; N=9, p=12; N=12, p=6; N=12, p=l2. Under all these conditions the fifty graphs were successfully recognized through step 1 alone.

3.2 Discussion-
There are special cases, resulting in failures, which raise an interesting question: what kind of local minima can be reached, and how difficult is it to escape from them and retrieve a ground state? A good illustration is observed when one presents an isomorphism of G_4 as an input. In 9 runs out of 50, the net with N=12 fails to identify G_4, whereas it learned G_1 to G_9 or G_1 to G_{11}. In 6 failures out of 9, the best match found by the system was G_2. This is not surprising since G_4 is a subgraph of G_2. Consider, for example, a rotated version of G_4 as the input (Figure 4). To escape from this state, three two-point exchanges that increase the energy should be accepted in sequence; conversely, if four-point exchanges were allowed, only one energetically unfavorable elementary move would be needed; if eight-point exchanges were allowed, this state would no longer be a local minimum.

The above example shows a particularly difficult case, because the input has eight possibilities to become a subgraph of G_2 and only two possibilities to match G_4. These results do not depend on the number of stored prototypes (within the range of parameters in our simulations), provided no new supergraph is introduced; however, if G_{12} is stored, then isomorphisms of G_4 are likely to be identified as G_{12}. Again, there are eight possibilities for G_4 to be a subgraph of G_{12}
There are also local minima that do not correspond to supergraphs, but have very small optimal distances. In the above example, G_5 is a good candidate, with $D^2(G_5, G_4) = 2$.

3.3 Recognition of non-isomorphic graphs
In these experiments, the unknown graphs are isomorphisms of one of the prototype graphs, with extra edges or missing edges. We studied in detail networks with N=12 and p=6, 9, or 12. The error-correcting properties of the system depend strongly on the number of stored prototypes. If 6 prototypes are stored, adding 3 or 4 edges leads to 100 % correct recognition. If 7 edges are added, 80% success is still achieved. An example of error correction is

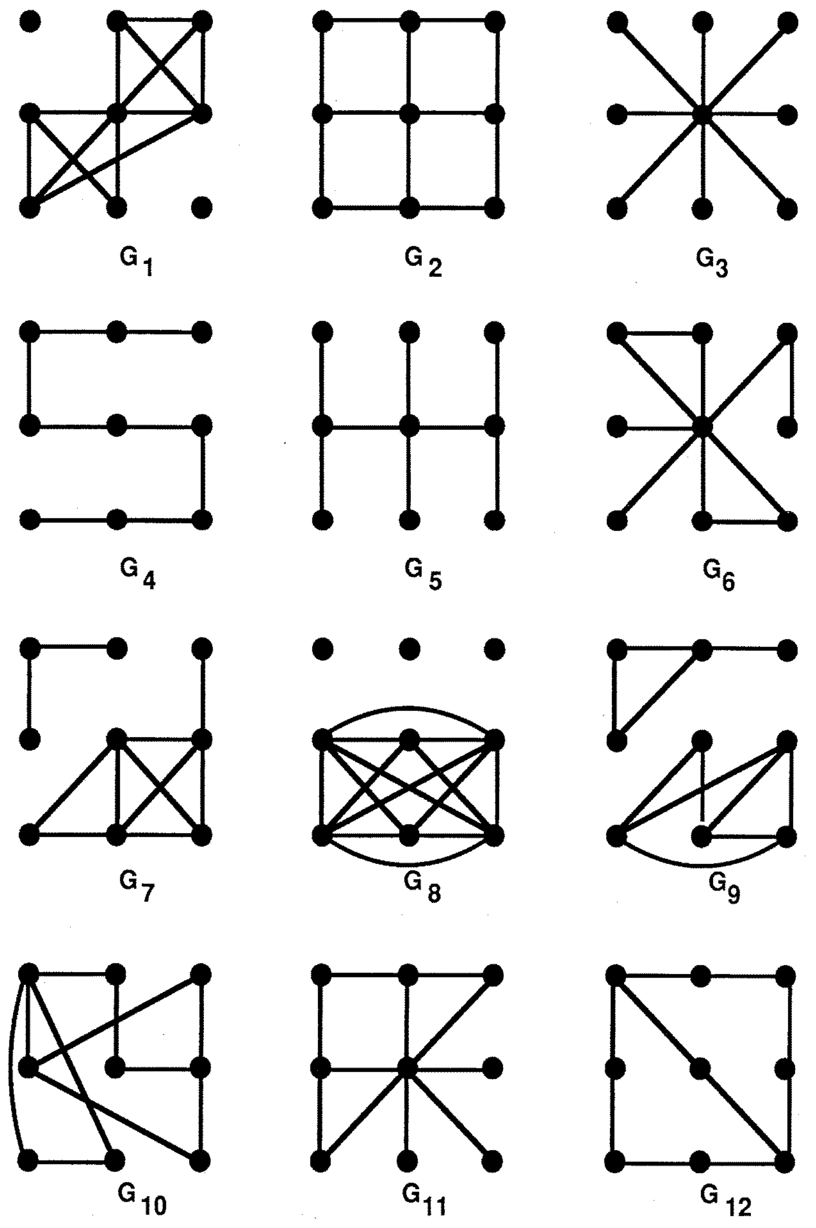


Figure 3
Twelve graphs stored as fixed points in the memory network.

shown on Figure 5. As expected, error correction becomes less efficient it the number of prototypes is increased this is due only in part to the classical saturation phenomenon observed usually in neural networks: it is also due to the fact that, as more prototypes are added, there is a greater probability that some of them are topologically close to each other. As an illustration, consider the following example: we stored the first 9 graphs in the network; it was presented with isomorphisms of graph G with edge 3-5 missing; the network identified them as G_7 in 7 cases out of 50. When we added an edge instead of deleting one, this happened 8 times out of 50: this is due to the fact that graph G_7 is the prototype graph that is closest (in terms of the distance between graphs) to G_1.

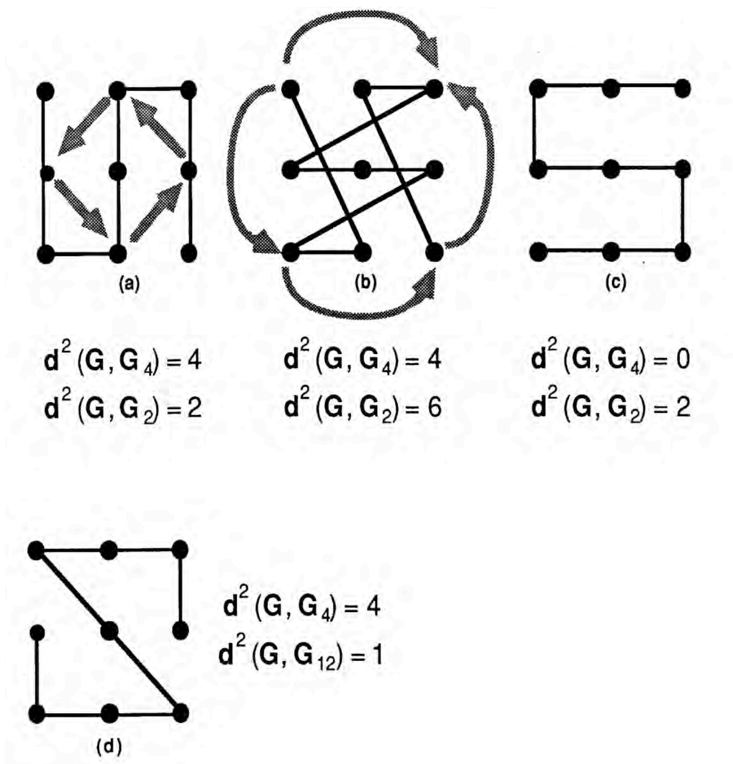


Figure 4
(a) A possible input G. The arrows indicate a four-point permutation leading to the graph shown in (b). This configuration represents a saddle point, from which the system can reach the global minimum shown in (c) by another four-point permutation. (d) A possible input G, which is an isomorphism of G_4 but has a small distance to G_{12}, which is a supergraph of G_4.

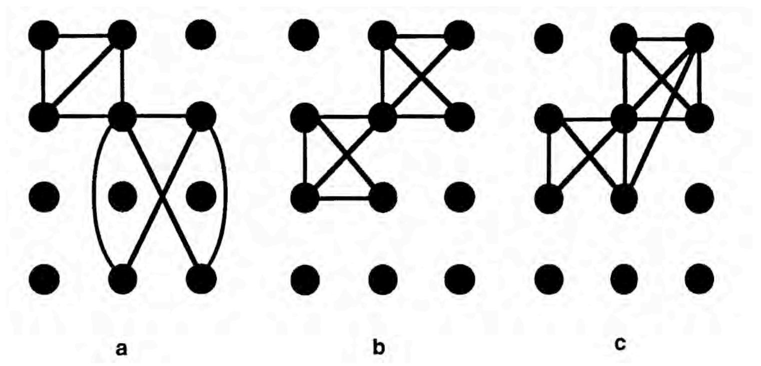


Figure 5
a) Input graph, which is an isomorphism of G1, with 2 missing links. b) Stable state reached by simulated annealing. c) State reached by the memory network: prototype graph G1.

3.3 Conclusion

The above results show that the simulated network model can generalize quite successfully over equivalent topologies. Even when it fails to recognize the correct isomorphism, it nevertheless identifies a graph that is topologically close to it. In this sense, the very failures correspond to a generalization over topological features - albeit an incomplete one. The rate of failures as well as the recognition times were found to depend strongly on the prototypes and on the input. A systematic assessment of the capabilities of the system will be possible only by investigating special classes of graphs.

4. CONCLUSIONS

The present paper shows that neural networks have the ability to handle structured data, so that they can be used for high-level processing; we prove experimentally that graph recognition can be achieved this may have an impact in various fields where graph recognition is important picture processing, scene analysis, description of chemical structures, relational database systems, switching theory, etc. It must be pointed out, however, that the procedure which is presented here is by no means optimized, and that a number of variations and improvements must be investigated among these are the use of alternative optimization conditions (exchanges of more than two nodes, constant-temperature operation), alternative optimization algorithms (is simulated annealing the most appropriate search method for this problem?), and alternative learning rules.

Furthermore, the simulations that were performed were restricted to a special case of a more general network for graph recognition; this choice simplifies the simulation to a great extent and makes error-correcting isomorphism possible, but it may not be optimal. The performances of the system should be investigated in the full space of the parameters that specify the architecture. It should also be noticed that the problem of graph recognition is treated here in a completely general way: the method can be applied to any non-labeled graph subjected to any topology-conserving transformation. For practical applications, this may be too general: some special classes of graphs, or some special transformations only may be of interest However preliminary the present results, the approach suggested here is promising because it may be expected to open a new class of applications for neural networks.

ACKNOWLEDGEMENTS

We would like to thank D. Amit, A. Treves and R. Kree for many interesting discussions; part of this work was carried out at the Institute for Advanced Studies at the Hebrew University of Jerusalem, to which we are very grateful for its hospitality.

This work was supported by the BRAIN (Basic Research in Artificial Intelligence and Neurocomputing) initiative of the Commission of European Communities.

REFERENCES

1 See for instance: K.S. Fu, *Syntactic Pattern Recognition and Applications* (Prentice-Hall, 1982)
2 R. Kree, A. Zippelius, J. Phys. A 21, L813 (1988).
3 J. J. Hoptield, D. Tank, Biol. Cybernetics 52, 141 (1985).
4 J. J. Hopfield, Proc. Nat. Acad. Sci. 79, 2554 (1982).
5 L. Personnaz, I. Guyon, G. Dreyfus, Phys. Rev. A 34, 4217 (1986).
6 L. D. Jackel, R.E. Howard, H.P. Graf, B. Straughn, J.S. Denker, J. Vac. Sci. Technol. B 4, 61 (1986).
7 Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Science 220, 671 (1983).
8 P. Siarry, L. Bergonzi, G. Dreyfus, IEEE Trans. on Computer-Aided Design 6, 211 (1987).