Hopfield Neural Networks

PROCESSOR ARRAY REQUIREMENTS FOR ADVANCED IMAGE PROCESSING: THEORY AND EXPERIMENT

M R B Forshaw

Department of Physics and Astronomy
University College London
Gower Street, London WC1E 6BT

ABSTRACT

Two topics are described. The first is the use of connection or reachability matrices as pictorial aids in the investigation of fixed-geometry processor arrays. Matrices are shown for mesh and hypercube arrays and also for a connection topology which is based on hashing. The second topic is an explanation why Hopfield-style associative memories exhibit large variations in the recall strength of individual patterns.

I INTRODUCTION

This paper is intended to provide a summary of some preliminary theoretical studies and computer simulations which have been carried out during the first phase of Alvey Project MMI/033. For a fuller description of the aims of this project, which is concerned with investigating the use of partially autonomous MIMD arrays for image processing, see Duff (1985). Much of the project is concerned with building and using a 256-processor array called CLIP7A (Fountain 1987).

The work described here covers two topics: (1) the investigation of new array architectures for image processing, and (2) the investigation of the variation in storage strengths of Hopfield-type associative memories.

II ARRAY ARCHITECTURES

Work in the first area is best described by a pictorial example. Suppose that we have an array of 256 processing elements (PEs), each with eight bidirectional I/O ports, and that we connect them in the form of a two-dimensional grid with eight-neighbour connections. Such a computer array would be a scaled-down 16 by 16 version of the 96 by 96 element CLIP4 image-processing array (Duff 1978). We can generate a Boolean *connection matrix* or *reachability matrix* with row and column indices **i** and **j** respectively, with the (**i**,**j**)th element equal to 1 if a connection exists from PE **i** to **j** and 0 otherwise. We are interested here in examining how fast information can be passed across the array. For one time step the connection matrix is the same as an adjacency matrix (Forshaw 1987a), but for two or more time steps the two matrices differ.

The connection matrix for this miniature eight-connected mesh array is shown in the first picture of Figure 1. Most of the rows have nine points, in three groups of three: the central triplet corresponds to the connection of each PE to itself and to its immediate left and right neighbours. On each side, at a spacing of 16 matrix elements along each row, a triplet of elements represents the PEs in the row of the mesh which are immediately above or below the chosen PE. The small scale of the picture obscures the fact that each of the diagonal lines is actually three elements wide.

The second picture in Figure 1 shows which elements can be reached in two steps or iterations. As one would expect, any one row contains five groups, each with five elements, corresponding to the 5 by 5 neighbourhood which any CLIP4-like PE can access in two steps. The third picture is for four steps or iterations: each row has nine groups, each with nine elements, corresponding to a 9 by 9 neighbourhood. After eight steps (the fourth picture) we would expect any one PE of an eight-connected CLIP4-like array to have access to a square neighbourhood whose side is of length (2 x 8 + 1) = 17. We see that the central PE of the 16 by 16 array (the row halfway down the picture) does indeed have access to all of the other PEs. Elements at the top of the array cannot propagate their information to the bottom of the array in only eight steps (assuming no wrap-around) and the top-right and bottom-left corners are therefore empty.

AVC 1987 doi:10.5244/C.1.17

If we were to generate a connection matrix for CLIP4, which has 9216 PEs, then the matrix would have dimensions 9216 by 9216. After eight steps this matrix would still be very sparsely filled, unlike the last matrix in Figure 1.

Other examples of connection and adjacency matrices are given by Forshaw (1987a, 1987b). We show here only one other example, for a 256-element array with hypercube-connectivity. Examples of such arrays are the Connection Machine (Hillis 1985) and the Intel iPSC series (Mokhoff 1985). The four pictures in Figure 2 show where paths exist between PEs in one step, one *or* two steps, one *or* two *or* three *or* four steps, and one *or* ... eight steps respectively. As one would expect, it is possible to pass information from any one PE to any one of the other 255 PEs in 8 (=$\log_2 256$) steps.

In one sense the connection matrices are no more than graphical representations of well-known Boolean matrix formulae. However, in another sense they provide a useful insight into which sort of architecture one might use for any given class of image-processing problem. It scarcely needs pointing out that the computer architecture which is best suited to implement a specific algorithm is one which has the same structure, in a suitably defined way, as the algorithm. If we were to leave considerations of cost or practicality aside then the best architecture would be either a completely reconfigurable array or else one with all-to-all connections. The former has been the subject of much theoretical and practical research (Kuehn et al. 1985). The latter is currently infeasible for more than about 64 PEs: experimental chips are reported to have been made with 54 elements having all-to-all connectivity on-chip (Anon. 1986).

Figure 3 shows the matrices for a randomly-connected array with eight connections per PE. The reader is reminded that a randomly-connected array provides faster one-to-any communications than the mesh, hypercube, or many other architectures. Its fundamental limitation is that the routing of labelled data is extremely difficult. We are exploring architectures which approach the graph-theoretical Moore bound for the fastest possible inter-element communication, while still permitting relatively easy data routing. These architectures are based on certain small-diameter, low-degree graphs, for which the name 'hash graph' is suggested as a suitable description. These graphs, or close variants thereof, have been independently discovered by many authors. The method used is closely related to that described by Imase et al. (1985): it is very similar to the methods used to generate de Bruijn and Kautz graphs (Bermond et al. 1986). The technique is based upon the 'multiplicative congruential' method, which is a well-established technique for generating random numbers and also for hash-coding data which are to be stored in a database (Knuth 1973). The principles of the method are discussed in (Forshaw 1987c), but may be summarised in the following expression, which defines whether a connection is made from PE x to PE y:

$$y = (ax + c) \bmod N$$

where a is a suitably chosen multiplier which is relatively prime to N (the number of PEs) and c is an index which in the simplest case could take the values 1,2,3,...,d, where d is the number of out-connections per PE.

The practical consequences of this technique are illustrated in Figure 4, which shows the connection matrices for a hash-graph network of 256 PEs, with eight outputs and eight inputs per node. The matrices are shown for one, two and three steps, and it can be seen that any-to-any connectivity can be achieved (in this example) in only three steps, compared with the eight steps needed for a 256-element hypercube network such as that shown in Figure 2. In general, if N is the number of PEs, and if $d_{in} + d_{out} = 2d$ is the number of wires per PE then the maximum number of steps needed for any-to-any data transfer is $\log_d N + 1$. This can be a significant improvement over the $\log_2 N$ steps needed for the hypercube or the $2\sqrt{N}$ for the 2D mesh.

The reduction in the number of steps is achieved at the expense of a reduction in the number of alternative paths between nodes which other networks provide. The adjacency matrices for 2D meshes, pyramids and hypercubes show that there are typically several hundred different paths between elements which are three steps away from each other (Forshaw 1987b), whereas hash graphs have only one or a few paths.

Although hash graphs offer the potential of high-speed data transfer, this is accompanied by several effects which can complicate the design and operation of any computer array which might use hashed connections. One problem is that of ensuring that data addressing is economical for both hardware and software. A second problem is that the hash graph topology does not correspond to that of any well-known image-processing operation. A third problem is that of laying out the connections, either as a VLSI structure or, on a larger scale, as printed-circuit boards and backplanes. These points are still being investigated, but Figure 5, which is a representation of wiring nets for different architectures, illustrates some of the problems.

Each of the diagrams in Figure 5 shows the 'wires' for the first 32 elements of a 256-PE array. The wires are in groups of eight: each wire extends along the x-axis to join PE number 0,...,31 to one of the other 255 PEs. The first diagram is for the eight-connected mesh whose connection matrix is shown in Figure 1. The short wire lengths are indicative of the locality of connections in mesh-connected arrays. The second figure is the wiring diagram for the first 32 elements of a hypercube array: it shows the longer range of some of the hypercube connections. The third wiring diagram is for a randomly-connected array with eight connections per PE: the average connection length is $N/2$, or 128 in this example. The high wiring density emphasises the problems involved in providing long-range connections in hardware.

The fourth diagram is for a 'hash graph'-connected array. At first sight it looks little different from the random-wiring diagram. However, it is possible to relabel the PEs to produce the wiring diagram shown in the fifth diagram. This is topologically equivalent to the fourth diagram, but significantly different in physical layout. It is possible further to modify this wiring diagram if we make use of the fact that the eight connections per PE cannot be accessed simultaneously (at least, not without an eight-fold increase in PE size). Most existing systems (CLIP4, CLIP7A, DAP (Parkinson 1978), Connection Machine, etc.) have directional multiplexers which are integrated with each PE or with a small cluster of PEs. By physically separating the multiplexers and their associated PEs one can obtain a large reduction in the amount of inter-PE wiring or tracks: the last diagram in Figure 5 shows how much the wiring can be reduced by this method.

III ASSOCIATIVE MEMORIES

There has recently been a strong revival of interest in the possibility of using networks of threshold logic units for a variety of computational tasks. One potential application of such quasi-neural networks is as associative memories, which might be suitable for use in higher-level image-processing tasks.

One example of such a network was described by Hopfield (1982): it has been the subject of much investigation, largely because it is more amenable to quantitative analysis than many other systems. It is also of considerable theoretical interest, and it offers the possibility of extension to include many functions which would be useful in higher-level image processing.

Analyses have been made of the data storage capacity of such an associative memory (e.g. Forshaw 1986). It does not seem to have been pointed out that the basis of some associative memory networks (e.g. Lansner and Ekeberg 1985) is very closely related in principle to hashing, which is long-established as a useful technique in data-retrieval systems. It is of interest to note that the Connection Machine uses hashing for searching large databases (Stanfill and Kahle 1986), thereby effectively mimicking an associative memory.

The Hopfield paradigm for associative memories assumes that the p patterns which are to be stored are random, with a mean overlap q between pairs of patterns which is given by $q = N^{0.5}$, where N is the pattern size. Thus as N increases the overlap becomes relatively small. The weight matrix $\mathbf{W}$ is given by $\mathbf{W} = \mathbf{E}\mathbf{E}^{\mathbf{T}}$, where $\mathbf{E}$, with N rows and p columns, is the ensemble matrix of the stored patterns and $\mathbf{E}^{\mathbf{T}}$ denotes the transpose. This is an approximation to the ideal weight matrix $\mathbf{W} = \mathbf{E}\mathbf{E}^{+}$, where + denotes the pseudoinverse (see e.g. Kohonen et al. 1981, page 118). The evolution of the system is given by $\mathbf{s}_{t+1}^{\mathbf{T}} = T(\mathbf{W}\mathbf{s}_t^{\mathbf{T}})$, where $\mathbf{s}$ is the system state vector at times t and t+1 and T is a nonlinear threshold operator. Several theories exist which provide, to varying degrees of approximation, predictions for the number of patterns which can be stored in such quasi-neural networks. Most of these are based on 'mean field' assumptions about the perturbations which are produced by the overlap of the weighted connections for any one pattern with those of all the other stored patterns. The resulting predictions about memory capacity do not therefore take into account the fact that there are indeed variations in the mean field. Statistical fluctuations ensure that some stored patterns have very small overlaps with each of the other patterns, while other patterns have very large overlaps. The effect of this is that some patterns can be reconstructed with considerable efficiency, even when only a small piece of the pattern is shown to the network. Other patterns can be reconstructed ('recognised') only by showing almost the entire pattern to the network.

An approximate measure of this variation in pattern recall strength is given by the (linear) eigenvalue spectrum Λ of the weight matrix $\mathbf{W}$, viz., $\mathbf{WE} = \Lambda\mathbf{E}$. If the patterns were truly orthogonal then Λ would, if suitably ordered, have a constant magnitude for p terms, then zero thereafter. The existence of overlap or correlation between the stored patterns produces a change in the eigenvalue spectrum. Patterns which have large eigenvalues have large basins of attraction: patterns with small eigenvalues are weak attractors. Figure 6 shows an experimentally-obtained ordered eigenvalue spectrum for a network of 64 nodes, with $p = 30$ stored patterns: a large spread in eigenvalues can be seen. A detailed theory is still in preparation, but a first-order expansion of the state evolution equation (with thresholding neglected) yields the following expression for the magnitude of the βth eigenvalue:

$$\lambda_\beta \approx N\left[1 + \frac{1}{2N}\sum_{\substack{\alpha=1 \\ \alpha\neq\beta}}^{p} C_{\alpha\beta}\right]$$

where $C_{\alpha\beta}$ is the matrix element for the overlap between the αth and βth patterns, which one may assume to be normally distributed. The solid line in Figure 6 is derived from this expression and shows modest agreement with the experimental eigenvalue spectrum. One may note that the Wigner semicircle eigenvalue distribution (Mehta 1967) does not seem to agree with the experimental results. It is intended to extend the theory to cover systems with weight matrices which are not described by the Hebbian rule.

REFERENCES

Anon. ""Neuron" Chips Emulate Brain Cells, Hold Promise of Much Faster Processors." *Byte* 11 (November 1986) 9.

Bermond, J.-C., C. Delorme and J.-J. Quisquater. "Strategies for Interconnection Networks: Some Methods from Graph Theory." *J. Parallel Distrib. Comput.* 3 (1986) 433-449.

Duff, M. J. B. "Review of the CLIP Image Processing System." In *Proc. Nat. Comp. Conf.*, 1978, pp. 1055-1060.

Duff, M. J. B. "Partially autonomous MIMD Machines." Alvey Project MMI/033 Application Document, June 1985.

Forshaw, M. R. B. "Pattern Storage and Associative Memory in Quasi-Neural Networks." *Pattern Recogn. Letts.* 4 (1986) 427-431.

Forshaw, M. R. B. "Array Architectures for Image Processing. 1. Connection Matrices." Image Processing Group Report No. 87/3, Department of Physics and Astronomy, University College London, 1987a.

Forshaw, M. R. B. "Array Architectures for Image Processing. 2. Adjacency matrices." Image Processing Group Report No. 87/4, Department of Physics and Astronomy, University College London, 1987b.

Forshaw, M. R. B. "Array Architectures for Image Processing. 3. Minimum Diameter Graphs." Image Processing Group Report No. 87/7, Department of Physics and Astronomy, University College London, 1987c.

Fountain, T. J. "The Development of Array Architectures Embodying Partial Local Autonomy." In *Proc. AVC87.* Bristol, 1987.

Hillis, W. D. *The Connection Machine.* MIT Press, Cambridge, Mass., 1985.

Hopfield, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proc. Natl. Acad. Sci. USA* 79 (1982) 2554-2558.

Imase, M., T. Soneoka and K. Okada. "Connectivity of Regular Directed Graphs with Small Diameters." *IEEE Trans.* C-34 (1985) 267-273.

Knuth, D. E. *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison-Wesley, Reading, Mass., 1973.

Kohonen, T., P. Lehtiö and E. Oja. "Storage and Processing of Information in Distributed Associative Memory Systems." In *Parallel Models of Associative Memory*, G. E. Hinton and J. A. Anderson (eds.). Erlbaum Assoc., Hillsdale, NJ, 1981, pp. 105-143.

Kuehn, J. T., H. J. Siegel, D. L. Tuomenoksa and G. B. Adams III. "The Use and Design of PASM." In *Integrated Technology for Parallel Image Processing*, S. Levialdi (ed.). Academic Press, London, 1985, pp. 133-152.

Lansner, A. and O. Ekeberg. "Reliability and Speed of Recall in an Associative Network System." *IEEE Trans.* PAMI-7 (1985) 490-498.

Mehta, M. L. *Random Matrices and the Statistical Theory of Energy Levels.* Academic Press, NY, 1967.

Mokhoff, N. "Concurrent Computers Make Scientific Computing Affordable." *Computer Design* (April 1985) 59-60.

Parkinson, D. "An Introduction to Array Processors." In *Proc. 40th Diebold Conf.* Venice, 1978, pp. 63-68.

Stanfill, C. and B. Kahle. "Parallel Free-Text Search on the Connection Machine." *Commun. ACM* 29 (1986) 1229-1239.

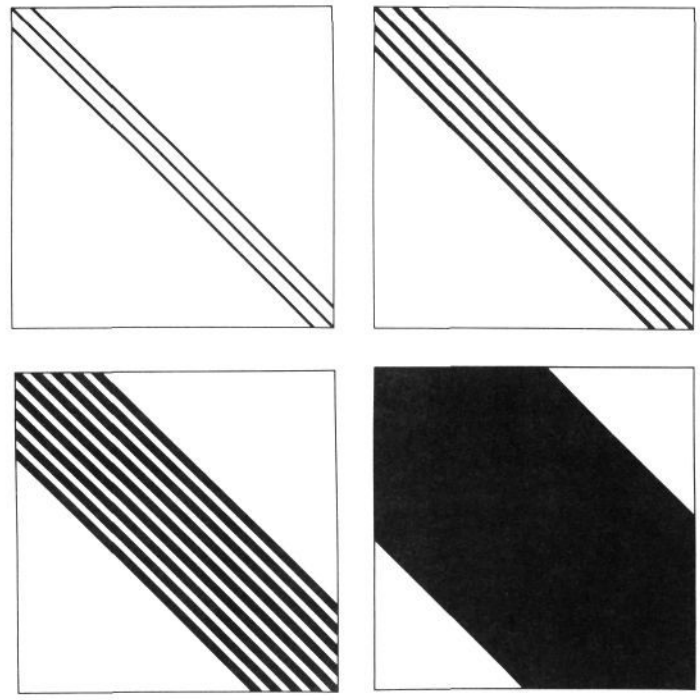

Figure 1. Connection matrices for a 256-element mesh-connected array with eight symmetrical connections per PE. The first matrix shows the elements which can be accessed in one step; the second, third and fourth matrices show the connections which can be achieved in two, four and eight steps respectively.

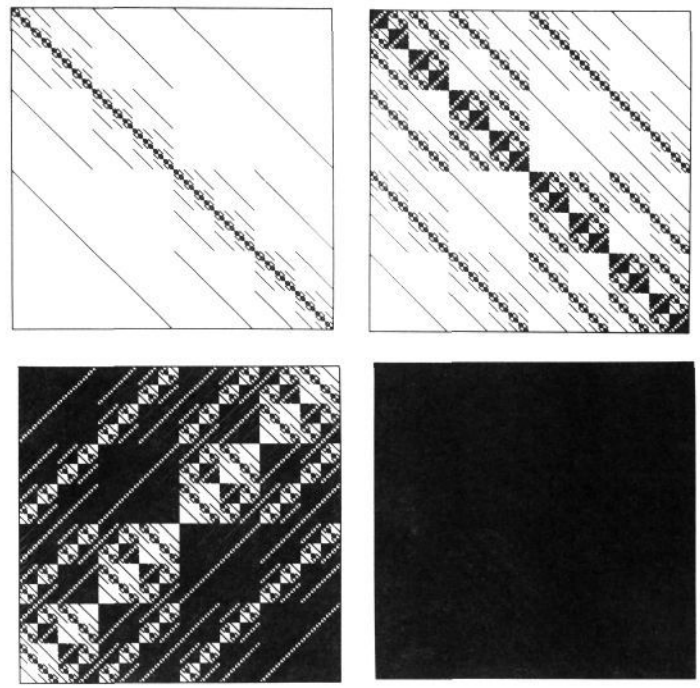

Figure 2. Connection matrices for a 256-element hypercube-connected array with eight symmetrical connections per PE. The first matrix shows the elements which can be accessed in one step; the second, third and fourth matrices show the connections which can be achieved in two, four and eight steps respectively.

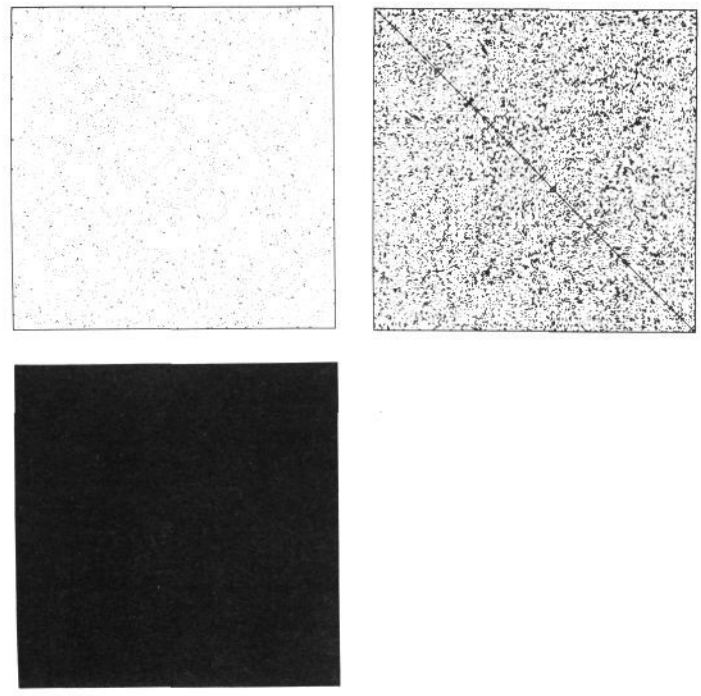

Figure 3. Connection matrices for a 256-element randomly-connected array with eight symmetrical connections per PE. The first matrix shows the elements which can be accessed in one step; the second and third matrices show the connections which can be achieved in two and four steps respectively. There are a few unfilled elements in the four-step matrix, but these disappear completely after five steps.

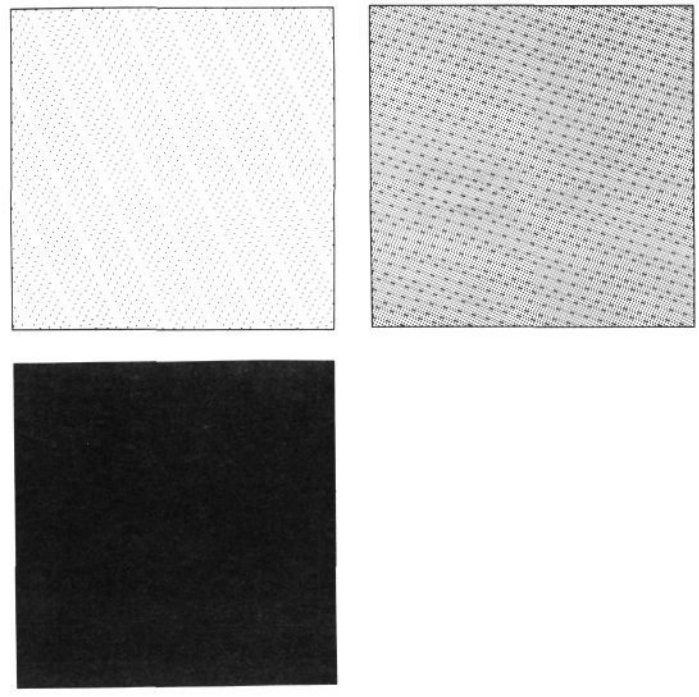

Figure 4. Connection matrices for a 256-element hash-connected array with eight asymmetrical connections per PE. The first matrix shows the elements which can be accessed in one step; the second and third matrices show the connections which can be achieved in two and three steps respectively.

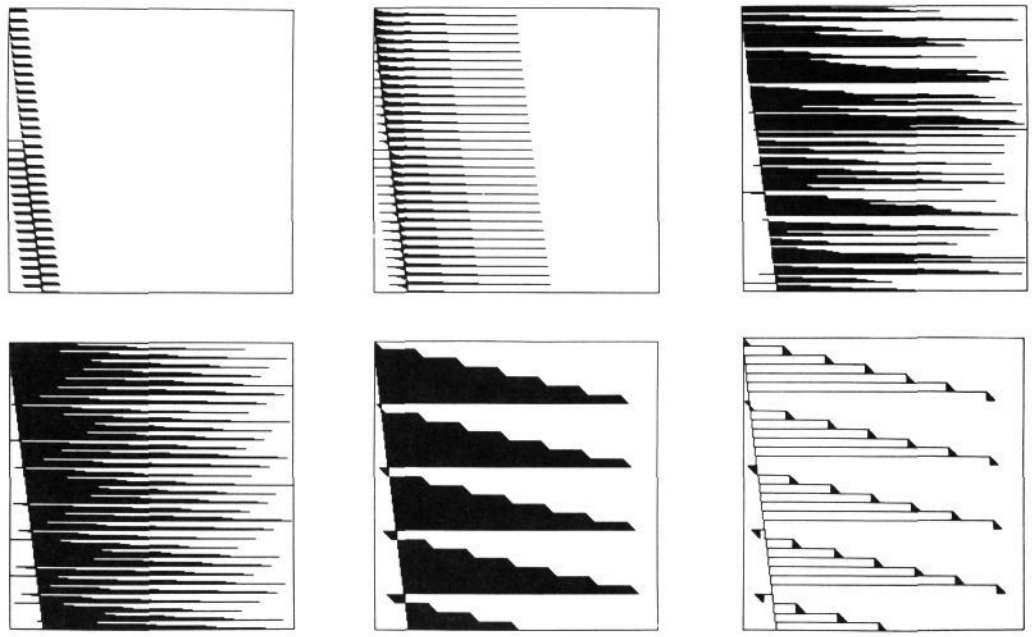

Figure 5. "Wiring diagrams" for systems with different topologies. Links are shown only for the first 32 elements of 256-PE arrays. Each horizontal line represents a unidirectional link, of which there are 8 per PE. The first picture is for a 8-connected grid, the second for a hypercube. The third is for a randomly-connected array and the fourth is for 'hash'-connections. The fifth is for a relabelling of the PEs in the fourth picture. The last picture is derived from the fifth picture by assuming that the data multiplexer for each PE can be separated physically from its PE.

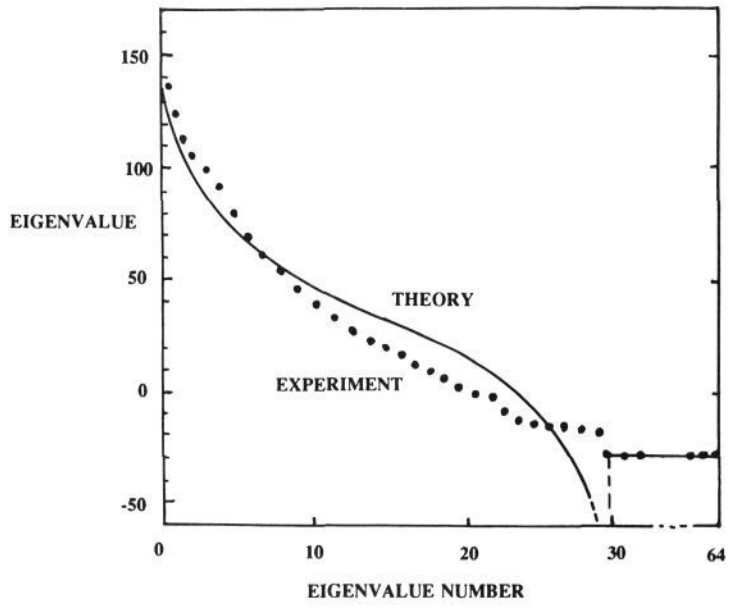

Figure 6. Measured and theoretical eigenvalues, versus ordered eigenvalue number, for an associative memory with 64 nodes and 30 stored random binary patterns. A pattern's recognisability is approximately proportional to the magnitude of its associated eigenvalue.