Neural Networks for Object Recognition

Nicola Cross, Roland Wilson
Department of Computer Science,
University of Warwick,
Coventry

October 9, 1995

Abstract

The overall aim is to produce a vision system using artificial neural networks that is capable of working with whole images. To this end some major problems in Computer Vision are considered. These are questions of control, binding, multiple pattern recognition, invariant processing and the modelling of complex visual behaviours. The majority of the practical work presented here is concerned with the development of higher order neural networks. In the higher order neural network, preprocessing creates a nonlinear decision surface for the network. A single layer will then often be sufficient for the network to train in a small number of passes. In the more traditional multilayered neural network, the hidden layers must extract the nonlinear quantities in the data for themselves. The number of passes through the data that must be made by the multilayer net for nonlinear problems is thus typically much greater. In addition, claims are made by some workers for superior robustness to noise and partial occlusion for the higher order networks. We also discuss and present some work pertaining to the control of vision systems.

Contents

List of Figures

1 Introduction

1.1 Perspective

The work described in this report forms part of a larger whole that is concerned with evolving a vision system using artificial neural networks (ANNs) that addresses some major issues in Computer Vision. These can be summarised as translation/scale/rotation invariant pattern recognition, representing repeated patterns (the 'banana' problem), network control and the binding problem.

Two areas are currently investigated practically. For the first of these, a neural network that is capable of distinguishing shapes at a coarse level of resolution is constructed. This provides the starting point for a search process which will begin with the identification of an object by its gross features, and proceeds to finer resolution levels as far as necessary for complete identification. The corners in the image are used as a means of identifying the shape of the object under consideration.

The second problem investigated is the control of the search process. Given that a search of an object database of any reasonable size is computationally demanding, the search should be optimised. This implies some measure of control so that the search can be made as efficient as possible. A general purpose vision system that considers multiple objects must be capable of aligning an object in a scene with stored representations of objects. An ANN that is capable of shifting an arbitrary one dimensional input pattern is constructed.

1.2 Binding

An important step in visual processing is the segregation of objects in a visual scene from one another and from the background. According to current theories of visual neuroscience the different features of a particular object are represented by cells which are spatially distributed across multiple visual areas in the brain. The segregation of an object therefore requires the unique identification and integration of the relevant cells which have to be bound into one group coding for this object. How this is to be accomplished is the so-called binding problem, and is currently a major issue in computer vision.

It has been suggested that such a binding of cells could be achieved by the selective synchronisation of temporally structured responses of the neurons activated by features of the same stimulus. This idea is supported by discoveries of stimulus dependent oscillatory activity in the visual systems of the cat, pigeon and monkey. The work by Gray et al [6] on the visual system of the cat led them to conclude that the synchronisation of oscillatory responses of spatially separate regions of the cortex may be used to establish a transient relationship between common but spatially distributed features of a pattern. They also remark that the synchronisation of oscillatory responses may have a more general function in cortical processing because it is a powerful mechanism for establishing cell assemblies that are characterised by the phase and the frequency of their coherent oscillations. However, the mechanism for this temporal synchrony is unknown. Gray et al suggest that connections from other cortical regions are responsible for the synchronous behaviour.

Hummel and Biederman [8] construct a network that uses temporal synchrony in their ANN for object recognition. Objects are seen as being made up of geons - simple three dimensional shapes. They establish the attributes of a geon as a chain of Fast Enabling Links (FEL)s. Whenever an attribute becomes activated, the other attributes on that chain also become activated via the FELs. The problem here is that any other object in the scene that happens to share an attribute with the activated object will be accidentally activated. This is perhaps an acceptable risk in a symbolic type of network such as this where the coincidence would be rare, but the interference would become more serious with more distributed architectures where the attribute needs to be shared by many parts of the same scene. Clearly, Hummel and Biederman have not solved the binding problem with this network, though other workers claim to have constructed networks that successfully use temporal binding; for example, Schillen and Konig (1994) used binding by temporal structure in multiple feature domains of an oscillatory neuronal network.

On the same theme, there the problem of representing multiple patterns to be considered. Not only must the network cope with the fact that the same feature may be associated with multiple objects in the scene, but also with the fact that the *same set* of features may be associated with more than one object. This would occur when duplicate objects must be represented; when a bunch of bananas appears in the visual scene, for example. Phase locking meets some difficulties here, because the same phase locking that binds the bunch of bananas together could not serve to differentiate the individual bananas. One would have to postulate the existence of some system where the individual bananas each fired on its own temporal slot, and at some other time fired in synchrony with all the other bananas in the bunch. Such a system would seem to be unworkable because there would need to be multiple temporal slots for each object. Hummel and Biederman consider at some length the plausibility of temporal phase locking as the method used by real neurons to establish binding. Their calculations on the timing of such a mechanism indicate that the establishment of synchrony would have to be extremely rapid. It is unlikely therefore, that there would be time for a complex multiple temporal slot system to be operated. There are other difficulties associated with the idea of using temporal phase locking to achieve binding, but in view of the difficulties presented here alone, this possibility is not the only one considered.

It would seem that the only aspect of the appearance of an object in the visual field that can be relied on to identify it uniquely, is its position; this is because it is impossible to have more than one object at any one point in the visual field. For three dimensional systems this is obviously true (nesting objects such as Russian dolls might be seen as causing a problem, but need not if perceived as hollow). Two dimensional systems may be complicated by occlusions, but if two identical objects are positioned such that one completely occludes the other, the problem no longer exists because only one of the objects can be seen. If one of the objects is not completely occluded then it cannot occupy the same spatial area. From a features point of view too, this approach would seem to be promising; those features sharing the same spatial location *must* belong to the same object. From this argument, the logical path would be to construct a vision system such that features and objects are referenced via a retinal system. This would constitute the 'where' part of a What-Where vision system. The 'what' part of the system need not be concerned with the spatial location of the objects.

1.3 The Problem of Scale

Another major issue in computer vision is that of scale. The multiresolution pyramid is a well established method that that is used in image analysis for a variety of purposes. The main advantage of using a multiresolution architecture is that image data can be examined at many levels of complexity at the cost of a small increase in storage requirements. Typically, the pyramid is quantised into layers - a 'stack' of two dimensional images where each layer is one quarter the area of the one below it. The bottom layer of the pyramid is the size of the original image, so the size of the whole pyramid is $1 + \frac{1}{4} + \frac{1}{16} .... \approx \frac{4}{3}$ times the size of the original image. This type of pyramid has a quadtree architecture where an image element on any level except the first represents some subsampled value of the four image elements directly below

Burt and Adelson [2] used multiresolution pyramids to build their Gaussian and Laplacian pyramids. Bhalerao [1] used quadtrees for multiresolution image segmentation; boundaries and regions are estimated, then iteratively improved. Coarse features such as large regions are detailed higher up the pyramid than fine features such as boundaries, which are detailed at the lower levels. Clippingdale [4] used the multiresolution pyramid to restore clean images from noisy ones.

Humans perform scale invariance in a seemingly effortless manner, and we know that there are complex cells in the retina that respond to features at different scales. There is above all, a need for visual systems to compress and coalesce image data without losing detail. The multiresolution pyramid provides a ideal way of doing this.

For the process of shape recognition there is an added advantage of using a multiresolution approach, and this is associated with the fact that shape features change significantly over scale. What appears as a vertex at one level, for example, may well appear as a curve at a coarser level. Thus an object description changes qualitatively as well as quantitatively over scale. Since we do not necessarily know the scale of objects in the visual field, it is advantageous to have internal representations at multiple resolutions for the purpose of comparing them to objects in the visual field. It may be possible to use this information to help identify objects. That is, the representations at various scales could be combined and used as a *signature* that characterises the way that an object changes over scale. Alternatively, a method that performs multiresolution scaling of the objects in the visual field so that they might match some stored internal representation could assist the same end.

In the ANN constructed to perform object recognition here, only the very coarse levels of the pyramid are actually used. At the 8 by 8 level the image is just emerging from the 'blob' stage, and a limited number of vertices can be distinguished. This is therefore the most economical point at which we can begin to distinguish shapes on the basis of their corner response. The images are subsampled and processed using a corner detector developed by A.King [9].

1.4 Active Vision

A topical and interesting approach comes from the Active Vision frontier. Active vision systems can interact with the environment; that is, they can move about in space to collect information, rather than being dependent on the information that is fed to them. Rao and Ballard [12] describe such a system that is based on what they call iconic representations, which are actually small visual templates. Their system uses a combination of the two multiresolution methods described above as part of a What-Where vision system.

One part of the system takes images from the outside as input, and characterises them in terms of their responses to nine orientated Gaussian derivative basis filters, at all levels of a five level multiresolution pyramid. This gives a 45 dimensional feature vector for each point sampled on the object. Scenes are first segmented such that an approximate boundary for each object is determined, then a small number of points can be chosen within this boundary. Rao and Ballard recommend using a method where points are picked from concentric circles centred at the most salient regions of the object (saliency being determined as points with high spectral power). Object identification is achieved by a brute force search of the object database for the stored feature vectors that most closely match those of the object currently under consideration. Stored feature vectors represent the object in a variety of rotations. Matching will generally succeed even when considerable noise is present, so long as the object is represented in the database from a reasonably close viewpoint. The matching process is robust because one of the favourable properties of high-dimensional feature vectors is that there is only a very small probability that an unrelated vector will be similar enough to cause confusion. The question of scale variation in active vision systems arises principally because of changes in camera position which give rise to 'looming' effects. Should a match fail in the first instance because of scale changes, then the most promising candidate can be subjected to a process which involves 'sliding' one feature vector across the other in the hope that a match will occur. The other part of Rao and Ballard's system is concerned with finding a point of interest in the image.

A major problem with this system is that there is no representation of objects as entire entities. The piecemeal representation provides no characterisation of the shapes of objects. It is therefore improbable that two objects with identical shapes but with different markings would produce similar feature vectors. All that can be expected from this system is that it will be able to identify objects *identical* to the ones represented in its database. One might imagine however, that this sort of representation might be suitable for identifying textures if the area scanned for responses were representative of the surface. One of the main advantages of using ANNs is that learning can be generalised to novel data, so that even if a trained system has never encountered a *particular* object before, it can make an 'intelligent' response based on its previous experience of *similar* objects. But this can only occur if the patterns requiring similar responses also have similar representations. The representational scheme used here would preclude generalisation to novel data. Another disadvantage of this representation is that in the absence of any grouping of representations according to type, brute force is the only strategy that can used for searching the database. The amount of time taken to do this would be prohibitive with an object database of any appreciable size. The problem of directing and controlling a matching process is part of the more general problem of controlling processes within ANNs.

1.5 Control in Neural Networks

The whole question of control in a vision system using ANNs is a difficult one. There has been considerable work done on the formulation of knowledge bases using fuzzy neural networks, but these are concerned with the representation of production rules, see [18] for example. There is also a whole area of work in the control of ANNs, but most of this has been developed with the aim of the automatic control of plant, for example, the work done by Psaltis et al [11]. This is not really relevant to the control of information and processes *within* the visual system. More akin to this need are the various Adaptive Resonance Theory (ART) networks developed by Carpenter and Grossberg, for example [3], and those which have been further developed by other workers. These use a system of interacting attentional and orientation subsystems which are aided by the use of 'reset' waves. These are generated when mismatches between input features and stored category information occur. The absence of the reset wave indicates identification of an object.

Many vision systems adopt a sequential approach to perception where each object in an image is subjected to the identification process in turn. But for visual systems that are capable of the perception of the contents of the whole visual scene *simultaneously*, there is a need for some way of maintaining the results of previously identified objects. This problem is related to those of representing multiple and repeated patterns that were discussed earlier. It would seem reasonable therefore, to postulate the presence of some 'storage' area for this purpose. Studies of the ability of monkeys to maintain visuo-spatial states in the pre-frontal cortex [5] might seem to support this idea, though it should be made clear that this work is based on studies of working memory which is distinct from object recognition. In fact, the principal sulcus receives signals from the posterior parietal cortex, where the brain processes spatial vision. The interesting point here is that certain neurons seem to posses 'memory fields': when a particular target disappears from view, a neuron switches on, and this neuron always codes for the same, sole location. It is interesting too that there is an adjacent area of the prefrontal cortex that contains neurons that respond preferentially to certain complex attributes of remembered objects. One neuron here responded more to a remembered red circle than to a green square, for example. Kosslyn [10] reviews the considerable neuroanatomical, neurophysiological and behavioural evidence for the existence of separate neural pathways for *what* and *where* processing in the human visual cortex. From Goldman-Rakic's work, it can be seen that there is evidence that the What-Where strategy for processing visual information is continued further into the more 'cognitive' areas of the brain. The case for What-Where systems has already been made during the discussion on binding, with reference to the ability of such a visual system to *uniquely* identify objects. In the absence of some dynamic binding method such as temporal phase locking, it seems reasonable to postulate the presence of numerous 'where' elements so that object attributes are recoverable. In the context of control, it implies that the 'what' and 'where' parts of the system are controlled as separate entities. Some attention controller could control and integrate the interaction between what and where parts of the system.

To represent objects in a database in every possible position, rotation and scale would be prohibitively expensive in terms of space; object representations must therefore be shifted, scaled and rotated so that the stored representations can be aligned with incoming data. In the Experimental Work section in this report, an ANN that is capable of a one dimensional translation is presented - a simple pattern 'shifter'. This is the first stage in the development of a procedurally based ANN attention control system. The procedural element is important because it allows the search process to be performed economically. A coarse to fine search strategy is proposed that will match incoming data with a hierarchical database. In this way, the search needs to make only as many steps as there are levels in the multiresolution pyramid. Siebert and Eising [15] use this coarse to fine matching strategy in their scale-space recognition based on the retino-cortical transform. This system has been shown to successfully build a plausible recognition tree using three geometric objects (circle, square and triangle) each of three sizes and 2 orientations. It is not an ANN though, so specifying the control of the system is trivial.

Many researchers such as Treisman [17] believe that the human visual system creates 'feature spaces' preattentively, and that attention is directed toward them so that the features can be combined into objects. The eye makes subconscious movements several times a second; these are called 'saccades'. It is thought that the function of these saccades is to foveate parts of the visual field containing significant features, so that the feature groups at that location become bound together. The grouping of cells into feature spaces would occur according to similarity, proximity and common motion, as was first proposed by the Gestalt psychologists. A feature space might be an area of one colour, or of similar texture, for example. It is easy to see that the feature spaces could be combined using the same principles to form the description of an object. For the purposes of the system planned here though, only the corners of the object are to be used as the distinguishing features, so some mechanism for scanning the visual scene is necessary. Some attentional 'spotlight' system could be implemented for this purpose.

It is hoped that the proposed ANN network system will be able to learn as well as classify objects. An interesting part of Siebert and Eising's work is the way that 'learning' takes place through creation of tree nodes, and 'association' through correlation and cortical image merges. Like biological systems, the distinction between learning and classification are somewhat blurred. Presentations of data that are judged to belong to an existing category will result in learning reinforcement, whereas data that is currently unrepresented, will cause learning to occur. There are similarities here to ART networks where there is the same ability to recognise new data; this is important so that 'unlearning' does not occur. In a network environment, cascade correlation network (CCN) training methods could be used to achieve this kind of learning. When a CCN fails to produce a required response, it enters a special training phase where candidate hidden nodes are created. The node that is most effective in reducing the global error of the network is taken and added to the permanent network structure. This process continues until the required responses can be made. A parallel between the growth and development of neurons and networks with dynamic architectures such as CCNs can be seen here.

As a mechanism for distinguishing between classes at one level of the pyramid, Hopfield networks could be used. Hopfield networks are recurrent networks that will settle towards one of a number of steady states so that the steady state selected most closely resembles the input. Yang [19] created a hierarchical structure of Hopfield networks for edge detection and restoration. Between layers of the database pyramid, the direction of flow would be from coarse to fine, and would be directed laterally between each pair of levels according to the matching results of the upper layer. Ambiguous cases could be tested in parallel, leaving the final discrimination for the lower layers of the pyramid working at finer grain resolution.

Figure 1 illustrates how a system that incorporates many of the features discussed in this section might be arranged. Note that in the figure, the 'what' part of the system contains the multiresolution database. Because there are multiple 'where'

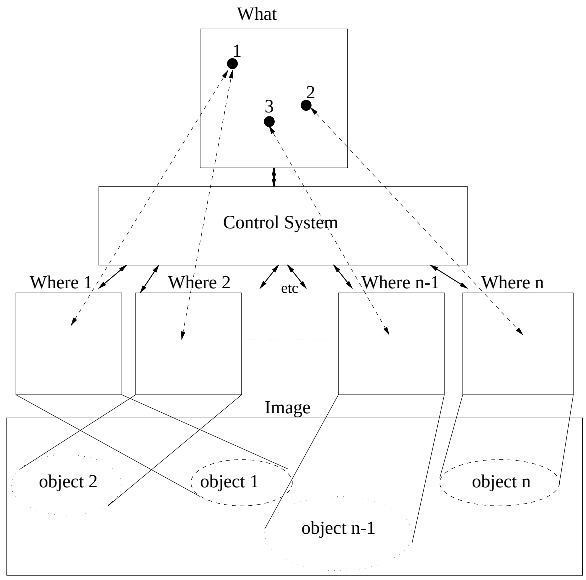

Figure 1: Proposed Layout for ANN Visual System

elements the problem of binding is overcome; it is always clear which 'what' belongs to which 'where'.

2 Theory

2.1 Artificial Neural Networks

The ANNs that are used in this work are feedforward networks that use a form of supervised learning called back-propagation (BP). These are well documented and extensively used. There is not space here to fully describe the workings of a neural network, so a brief description must suffice. The units are typically connected by links to other units, and each link carries an associated weight. The solutions are statistical ones where the solution arises out of a process of weight adjustment in an attempt to minimise the global error of the network. During training, the training data are presented to the network along with the ideal responses. The actual response of the network is compared to the ideal response, and the error is calculated. Adjustments to the weights are made by propagating the error backwards through the network using a gradient descent learning rule so that next time this pattern is presented to the network, the response is nearer the target. The process is repeated until, ideally, all training patterns produce a response that is within some preset error bound. Ideally too, the network is able to generalise its knowledge to novel data. That is, it is hoped that the network has learned the underlying principles used for grouping the data that will allow it to produce the correct response for data falling within the same category.

ANNs are particularly appropriate tools for this problem of visual shape recognition for several reasons. The networks are highly parallel, and since image data generally consists of large two dimensional arrays, the processing benefits from substantial savings in terms of efficiency - or at least they would if implemented on parallel machines. Here, the parallel algorithms are simulated on a serial machine. The similarities between multiresolution pyramid architectures and neural network architectures make the modelling of such pyramids using neural networks a straightforward task.

In terms of neural plausibility too, the ANN approach must gain some credit. Hubel's book 'Eye, Brain and Vision' [7] is devoted to a study of the construction and operation of the human visual system. ANNs show some similarity to biological neuronal systems in both structure and function. Because the human visual system is the only really comprehensive working model of a visual system available for study, it is both interesting and informative to work within a paradigm that has relevance to this.

A third advantage of using ANNs to study the vision problem is that there is no need to use heuristics to attempt to describe what it is that is significant in determining the characteristic features of shapes. The strategies employed for distinguishing shapes should appear automatically; that is, the network will formulate its own rules. Rosenberg and Sejnowski's NETtalk [13] which learned to pronounce English text provides a neat example of this. NETtalk managed to partition its hidden-unit vector space into 79 subspaces, one for each of the 79 letter-to-phoneme transformations that characterise the phonetic significance of English spelling. Since there are 79 distinct phonemes in the English language, but only 26 letters, it is clear that each letter can have more than one phonetic interpretation; the correct one is determined by context. Despite this ambiguity, NETtalk managed to learn which transform to select by being sensitive to the letters surrounding the target letter.

2.2 Higher Order Neural Networks

The HONN can be considered as a method for preprocessing that creates a nonlinear decision surface. These HONNS are a group of networks that incorporate domain specific knowledge into the network. One of the problems that has to be solved in computer vision systems is that of achieving invariance to various sorts of transformations of the image. With a first-order network, the traditional approach to solving the problem of object recognition is to present the network with many distorted views of objects, and allow the network to extract the invariant features by making many passes through the data. The HONN, on the other hand, typically requires only a very few passes through the data because the distortion invariance is built into the network and does not have to be learned. For a second-order network, input nodes receive values that are the products of *pairs* of data values - pixel grey levels for example. For a third-order network, the input nodes receive data that are the products of *triples* of data values. See figures 2, 3 and 4 for a comparison of the architectures of the first, second and third-order network architectures. The hidden units are not always present in first order networks.

In the second-order network, the pairs of data points may represent some known associations between input nodes. For example, Spirkovska and Reid [16] create a network that is invariant to distortions in scale and translation by constraining the weights of those pairs of input points that form the same *slope*. All pairs of points that form the same slope are connected to the same input node. What the input node actually receives is the *total* of all input pair products where all the input pairs have the same slope. Scaling and translation do not affect the slopes of the image points that characterise a shape, therefore an object representation that represents objects in terms of the slopes of their edges will be scale and translation invariant.

Spirkovska and Reid used a third order HONN in their work on distinguishing the shapes of various types of aircraft. In this, the input triples form triangles. Input triples that form similar triangles are connected to the same input node. What the input node actually receives is the *total* of all input triple products where all the input triples form similar triangles. It is the pattern formed by the summed products of the input triples that characterises a shape. The set is the same irrespective of the rotation of the shape; thus the network is rotationally invariant. Spirkovska and Reid state that the HONN not only trained faster, and generalised better with novel data, but was also more robust to noise and partial occlusions.

There are disadvantages in using HONNs and one of these is in the time and effort required to preprocess the data that the network uses. Another is that there are some problems caused by the number of product values that are generated by HONNs. The second-order network generates ${}_nC_2 = \frac{n!}{(n-2)!2!}$ products, and the third-order network generates ${}_nC_3 = \frac{n!}{(n-3)!3!}$. Even with an image as small as 16 by 16 the number of product values generated by a third-order network is over 2 million. There is a clear combinatorial explosion of weights as the image size increases. Spirkovska and Reid use a coarse coding method to reduce the number of input nodes required.

Another approach taken by Schmidt and Davis [14] reduces the connectivity of the network. This in turn reduces the storage requirements for weights. The reduction in connectivity is achieved by the use of partial templates as features. The features chosen might be: scene average (1×1), N adjacent pixels $(1 \times N)$, edge detecting features $(1, -1)$ or $(1, 0, -1)$. The sum of products of features is calculated for the entire scene, and this gives translation and rotational invariance. The number of network inputs equals the number of features plus a bias term.

In the following work, a novel method that effectively reduces the number of image points is implemented.

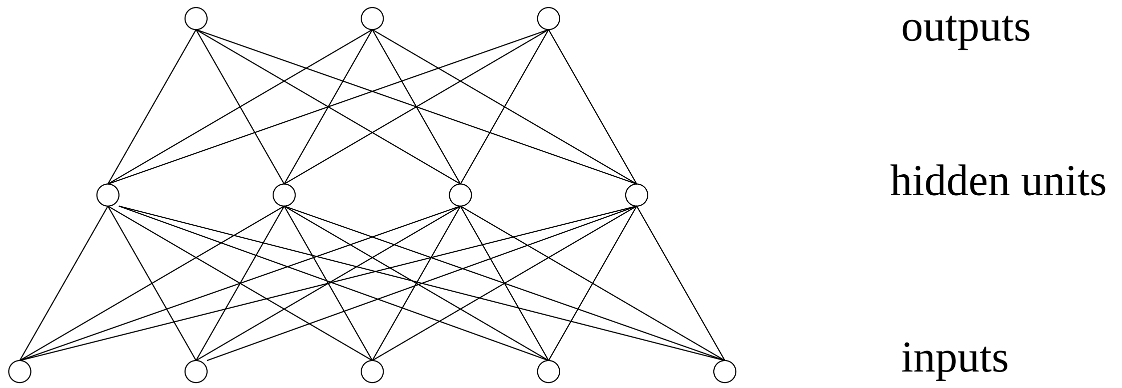

Figure 2: First-order network architecture.

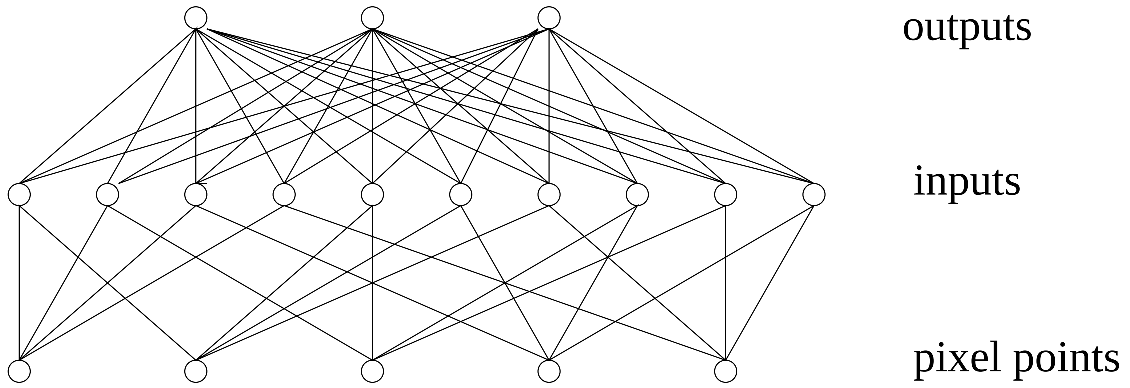

Figure 3: Second-order network architecture.

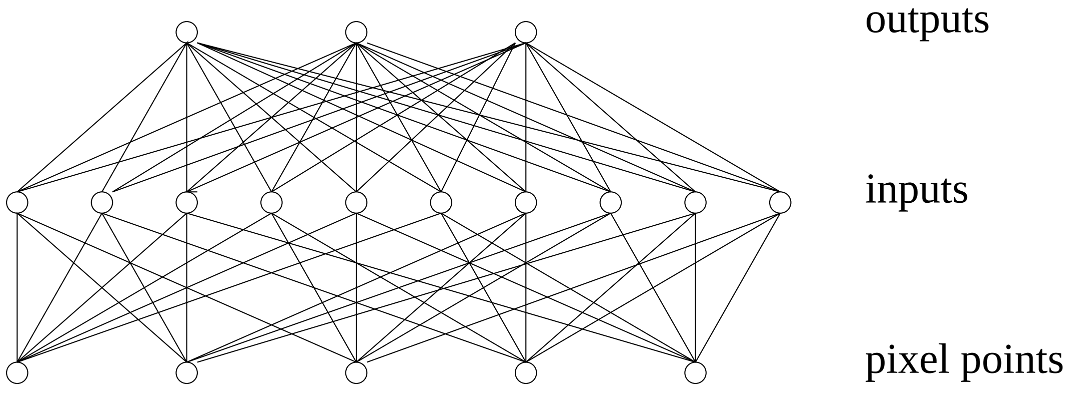

Figure 4: Third-order network architecture.

3 Experimental Work

In this section the results from several networks are presented. The networks were developed in series as it became apparent that there were difficulties inherent in some of the earlier models.

3.1 Data

The networks were trained on data which consisted of bars, triangles and squares in various sizes and orientations. The shapes are normalised according to their first moments; that is, each shape is positioned in the image plane according to its centre of mass. The shapes are normalised in size by their second moments. The size is allowed to vary upwards by about 50%. An increase in size greater than this would cause the shape to overlap the image boundaries. See figures 17, 18, 19, 20, 21, 22 for details of the data set. The testing data set consists of data which is similar to that in the training data set, though no two images are common to both data sets. Note that the corner response for the shapes in the data sets is shown as a greyscale value. In fact the response is available either as a set of scalar values, as shown here, or as a set of complex values. Each complex value contains two items of information: the modulus of the complex number gives the magnitude of the response whilst the argument gives the response at that point a *direction*. Corner response vectors will (ideally) point *into* the corners.

3.2 Second Order Networks

Second Order Networks were described in the Theory section. These were developed in an attempt to teach networks to produce a response denoting the object type (number of vertices) and the object orientation in response to the presentation of image data describing the image in terms of its corners. Feedforward networks using a backpropagation learning rule were used. The networks were constructed both with and without hidden units.

It was possible to teach the network to produce the object type, but not the orientation. It is always difficult to know exactly why a network will refuse to converge to a solution. The difficulty arises partly because networks are notoriously opaque; that is, it is difficult to see what is happening inside them. Another major complicating factor is the large number of parameters that have to be set, such as the learning rate and inertia parameters. The data set too has to be chosen carefully. One problem with this network in particular is that the the large number of inputs makes the search space huge. This means that the solution to a problem can 'swim about' in this 'bath like' architecture without any guarantee of success. Indeed, adding hidden units to the architecture actually interfered with the convergence of the network, presumably because the network architecture becomes increasingly amorphous with the addition of more units.

It was also apparent that the presentation of data as complex values made the task more difficult. In no case did the network converge to a solution when the complex data were presented. In this type of network, each input node receives a value that is the product of a pair of data values. For the complex data, the values presented to the network were the inner products of the vector multiplication. This will have caused problems because the inner product of two orthogonal vectors will equal zero, creating null spots where high values should be, at the product of two corners of the square, for example. A more satisfactory solution would have been to supply the results of the complex multiplication to a set of four input nodes, each of which would represent a part of the multiplication result. For example the result of multiplying the complex numbers $(a + ib)$ and $(c + id)$ is $(ac - bd) + i(ad + bc)$. The four input nodes would represent: ac, bd, iad and ibd.

Because of the problems occasioned in getting the network to converge, and because these were difficult to unravel with such an unstructured network, more closely constrained networks were defined.

3.3 Third Order Networks

The architecture of the third order network was discussed in the introduction. In brief, each input node receives values which correspond to the product of three data points. Again each data point represents the corner response. If the number of input nodes required for a second order network receiving its inputs from an 8 by 8 image is ${}_{64}C_2 = 2016$, then the number of input nodes required for a third order network receiving inputs from the same size of image is ${}_{64}C_3 = 41,664$. Since the large number of inputs seems to cause problems for the network, a method to reduce this was attempted.

To reduce the number of inputs to a reasonable level for the third order network, the input field is divided into 16 sectors - see figure 5 which shows how the image of a bar's corner response is split into the 16 sectors. The response in each sector is then summed to provide a per sector response value. This means that the number of effective image points is reduced from 64 to 16. The number of input nodes required for the third order network thus becomes ${}_{16}C_3 = 560$ instead of 41,664.

The networks had a single layer architecture and were constructed to receive either the 560 scalar inputs, or the 1120 complex inputs. The complex inputs were delivered as inner products as described above. The networks were trained and tested with the square and bar data sets as described for the second order networks. See figures 17, 18, 20, 21. Figures 6, 7, 8 show histograms of the summations for the sectors of bars, squares and triangles in five different orientations.

Similar training/testing results were obtained for this network as were gained from

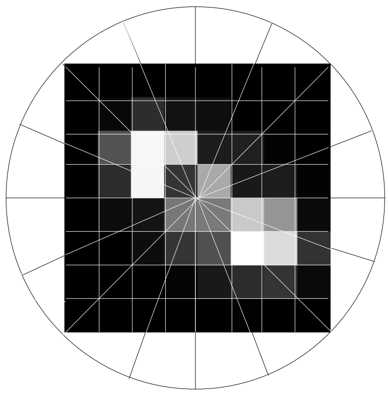

Figure 5: Splitting the image into sectors.

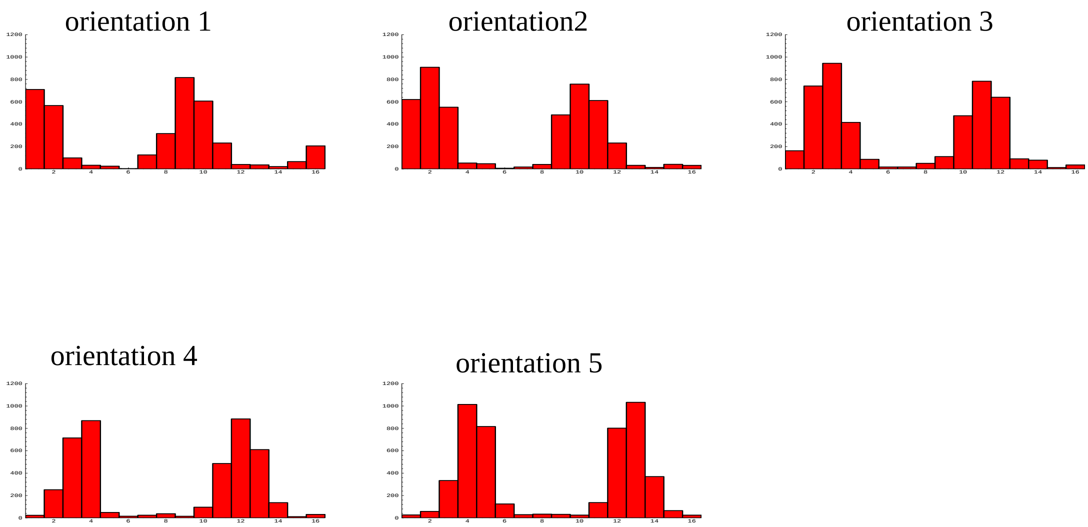

Figure 6: Sector response to a bar in 5 different orientations. The x axis denotes the sector number, the y axis shows the response.

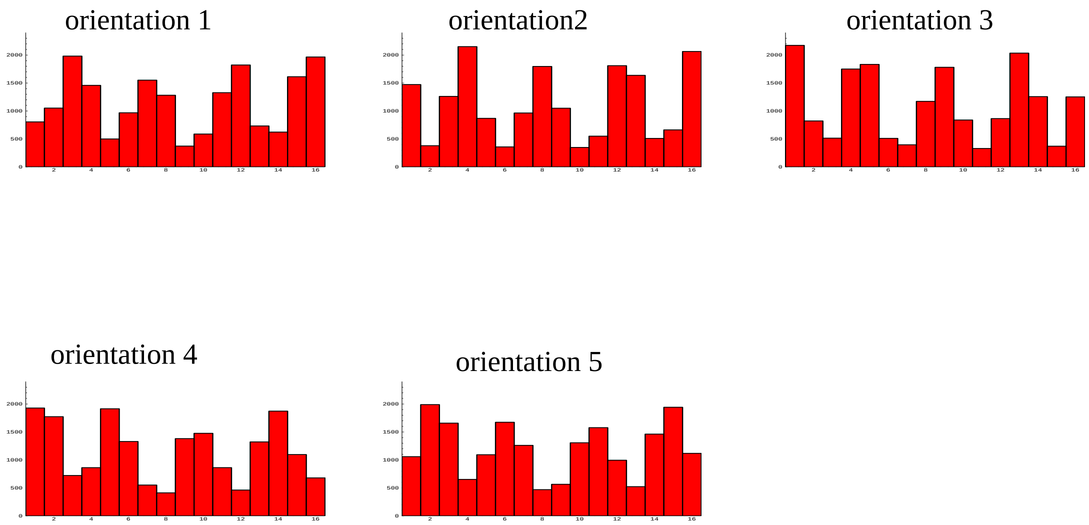

Figure 7: Sector response to a square in 5 different orientations. The x axis denotes the sector number, the y axis shows the response.

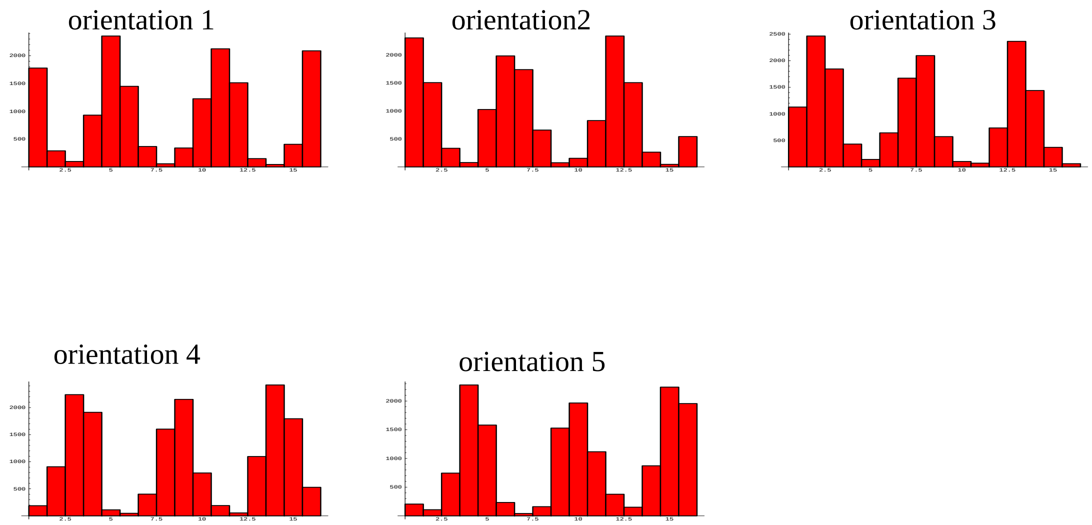

Figure 8: Sector response to a triangle in 5 different orientations. The x axis denotes the sector number, the y axis shows the response.

the second order networks. It was possible to obtain convergence on scalar data only; the complex data causing problems as before. For output, it was possible to obtain outputs of object type only; convergence failed when orientation was requested as part of the output. Some generalisation was evident when the testing data were presented to the trained network, but this was imperfect. In an attempt to improve the quality of the solution, a hidden layer was included in the network architecture. It was surprising to find that the network then refused to converge under any circumstances. Examination of the errors at various places in the network during training indicated that the error could be reduced to near zero for *either* squares *or* bars, but not both.

It was apparent that there were some difficulties associated with the complexity of the data which presented shapes in various sizes and in various orientations. It was decided to reduce the complexity of the data by preprocessing it to remove the rotational variation.

3.4 Third Order Networks with Rotational Invariance

Rotational variance is removed by presenting a data set in terms of the set of triangles formed by the third-order combinations of the sectors. This set of triangles will be the same irrespective of the orientation of the object present in the image. The method used is similar to that described in the Theory section for the work done by Spirkovska and Reid on the recognition of aeroplane outlines. A major difference between their work and that done here is that here the mid-points of sectors are used instead of image (pixel) points. Spirkovska and Reid use a coarse coding method to contain the problems posed by large numbers of inputs. In this method the same problem does not arise because the number of input points has already been drastically reduced by sectioning the image.

Figure 9 gives an example of one input triple and its rotational duplicates. For 16 sectors there are 560 input triples, and from these just 21 different triangles can be formed. 14 of these triangles have 32 duplicates as illustrated in figure 9, and 7 (the input triples that form isosoles triangles) have 16: $(14 \times 32) + (7 \times 16) = 560$. The third-order network requires one input node for each triangle with a unique set of angles; it therefore requires 21 input nodes. When the data is finally presented to the network, each input node will receive the *total* response for *all* input triples which form triangles with the same angles.

Figures 14, 15, 16 show histograms of the inputs to the network that were used for training. This time it can be seen that the peaks and troughs in the data stay in approximately the same positions despite the changes in orientation. The same *pattern* is evident, even though the *scale* of the pattern may change with the size of the original image. Note that for long, thin shapes like the bars used here, the heights of the peaks will vary substantially as the shape rotates. The reason for this is that a given sector will sometimes encompass the majority of the corner response in an

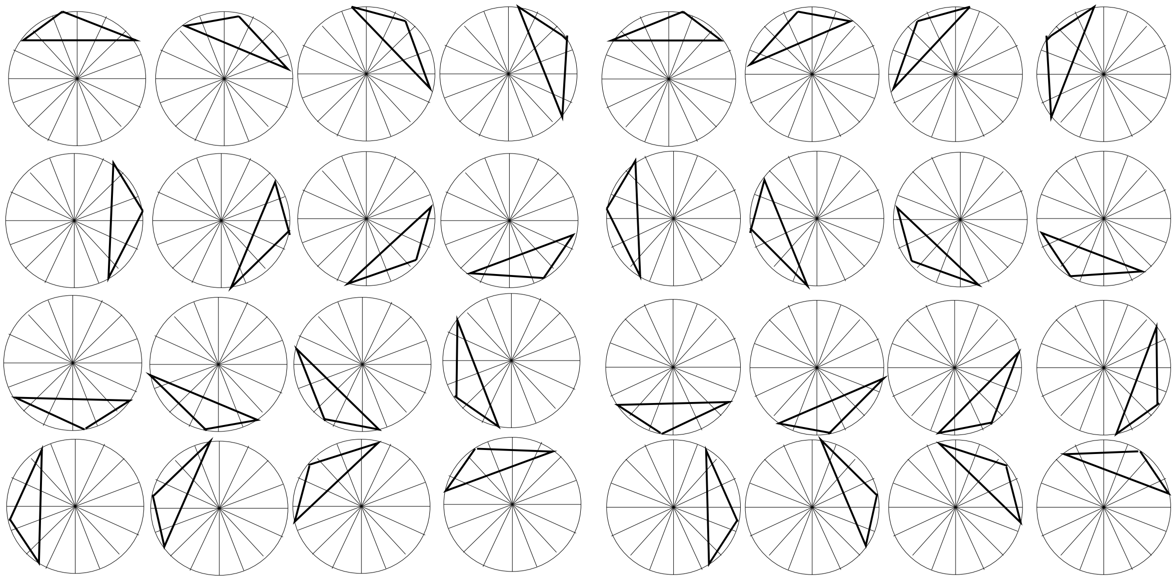

Figure 9: An input triple consists of the triangle formed by three sector points.

image; at other times this response will be spread over more than one sector. The effect is less noticeable for 'blob' like shapes because the corner response is more evenly distributed over the sectors. The task for the network is now simply to distinguish between the patterns for bars, squares and triangles.

A single layer network with 21 input nodes was trained on the scalar data produced from the data sets shown in figures 17, 18, 19 as before. The data shown in figures 20, 21, 22 were used as testing data as before. The network converged very quickly, in 27 epochs. All test cases were successfully classified to an error bound of 0.01. Figures 10 shows the results of noisy data on the convergence rates of the network during training. Note that the network will not converge when the SNR falls below 10. Figure 11 shows the results of testing networks trained with various degrees of noise on data which is also subject to various noise levels. The results are interesting: it would seem that at moderate noise levels (SNR 6 and above) the network that is trained with the noisiest data is best at classifying noisy test data, whilst at higher noise levels (SNR 6 and below) the complete reverse applies and the network that was trained with clean data performs best. However, the differences for percentage correct classification are not great, and all networks correctly classify all clean test images.

The question of producing the orientation as output no longer applies because inputs are now near identical irrespective of orientation. The failure of previous networks to converge is largely attributable to the fact that the weights were not constrained in any way. In the Theory section, methods used for constraining the

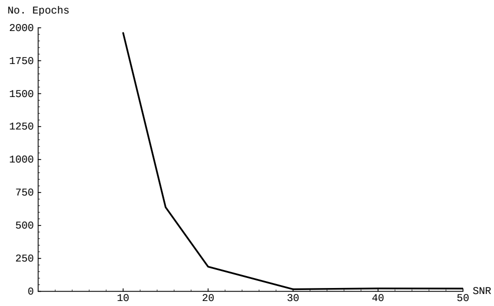

Figure 10: Effect of noise on convergence rates of the third-order network.

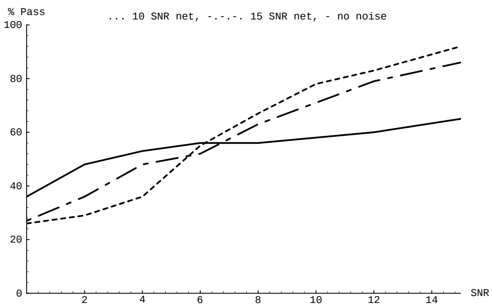

Figure 11: Effects of noise on the correct classification of test data with various amounts of noise added.

weights of a second-order and a third-order network were discussed. It is clear in retrospect that the HONNs without weight constraints are simply more complicated versions of the first-order network. Figure 12 demonstrates this. Four simple input patterns are shown at the top of the figure; these show a line in 4 different orientations. Below these appear the four input patterns that would be generated by second order products without weight constraint. The second form contains no additional information, and is longer than the first.

```
010  000  100  001
010  111  010  010
010  000  001  100
 a    b    c    d

000000000010010000000000000010000000   a
000000000000000000000110001000000000   b
000000000000000010100000000100000000   c
000100010000000000000000000001000000   d
```

Figure 12: Second order inputs for four simple input patterns.

The variation in peak height for bars that was mentioned above does not seem to be causing any problems for the network in distinguishing the three shapes. However it is possible that if more and different shapes were include in the data set, steps would have to be taken to reduce the amount of variation in triple input patterns. The number of sectors could be increased as one approach; this would reduce the variation of the maximum level of sector response as orientation changes. Additionally, it would be possible to include some radial averaging function in the preprocessing of the data so that the variation in the level of response that occurs because of differences in the size of the original image is removed. This variation is not apparent in the figures included here because to minimise the number of illustrations required the shapes shown are all at the same size.

The use of complex data caused failure as completely as before. Figure 13a shows the corner response for a bar with its long axis orientated at about 72^o anticlockwise from the positive direction of the x axis. Figure 13b shows the sector response for the same bar. In figure 13c the magnitudes of the vectors have been equalised so that all phase components are equally visible. From this, it is clear that the image background (ie: the places in the image where there is a low corner response) contains vectors that do not point towards the corners. It is possible that the lack of consistency in the corner response as the distance from the corner increases is confounding the problems already discussed for the use of complex data.

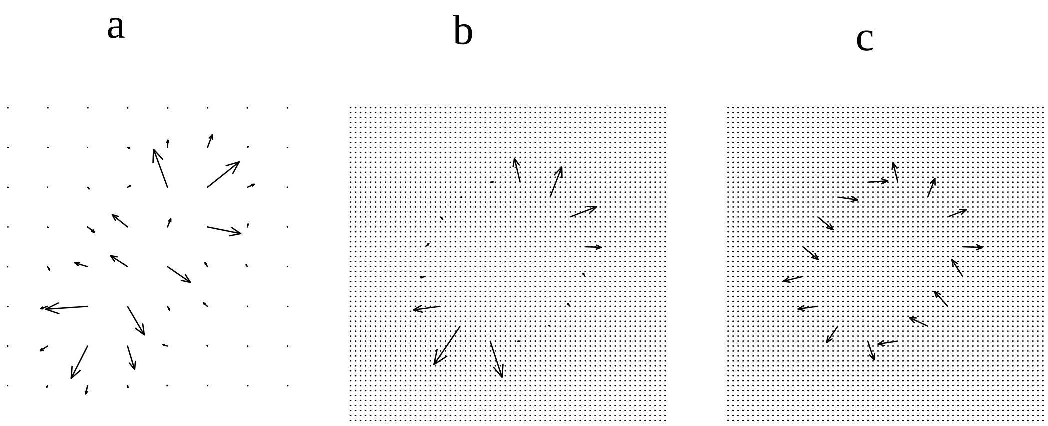

Figure 13: Vector plots of corner responses to a bar. Arrows indicate the direction of the corner response. The size of an arrow reflects the magnitude of the response at that point, except in c (see text). The background spots are not significant.

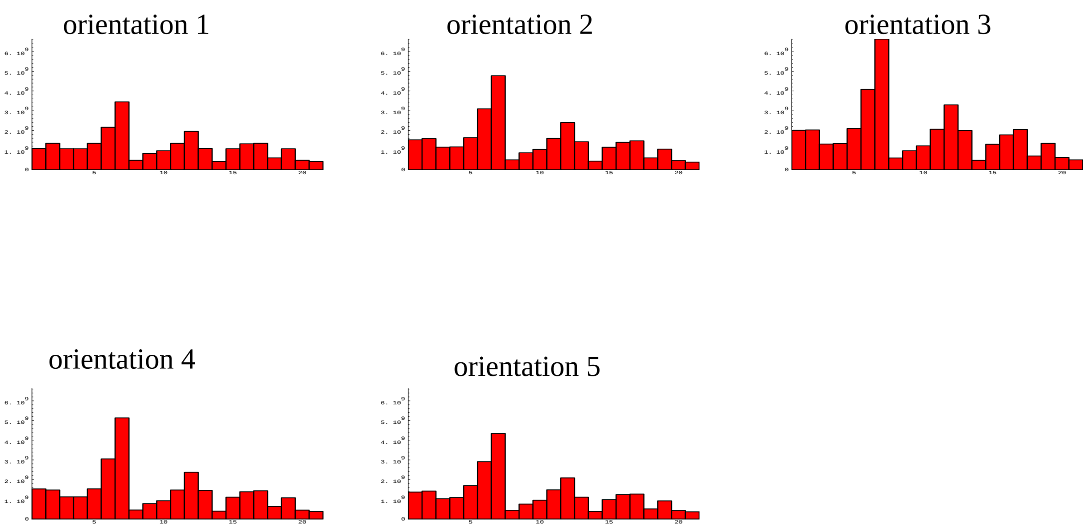

Figure 14: Network inputs for a bar in different orientations. The x axis denotes the input node number, the y axis shows the total input.

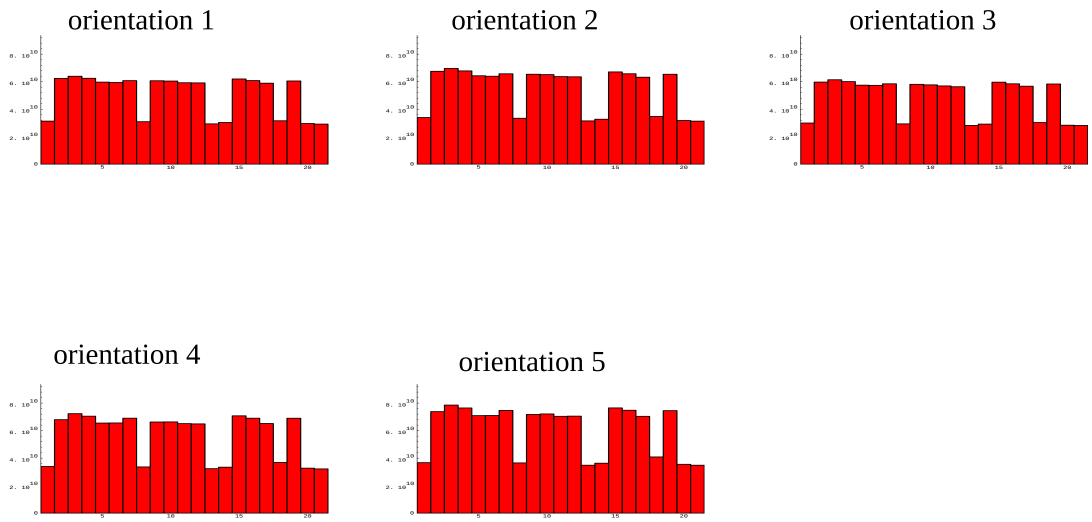

Figure 15: Network inputs for a square in different orientations. The x axis denotes the input node number, the y axis shows the total input.

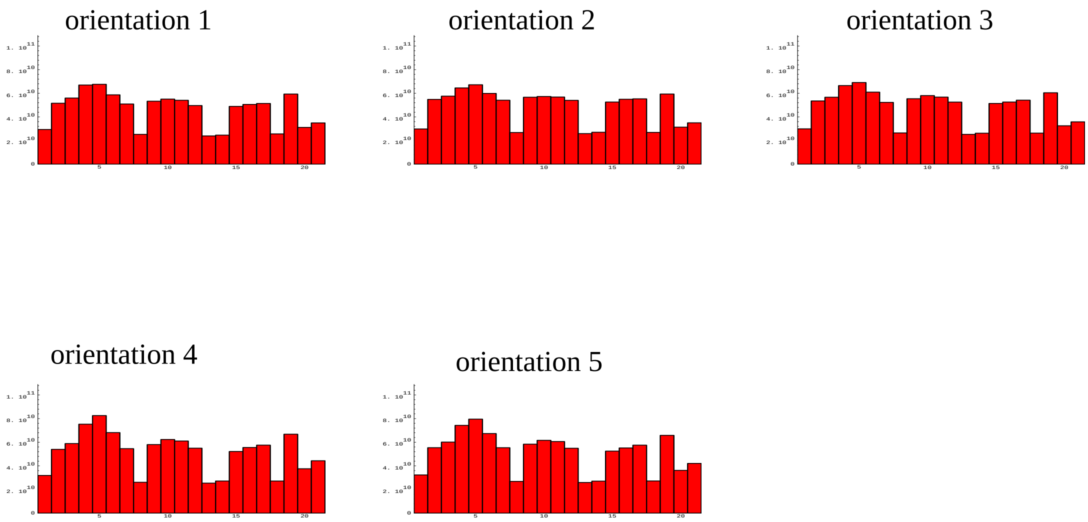

Figure 16: Network inputs for a triangle in different orientations. The x axis denotes the input node number, the y axis shows the total input.

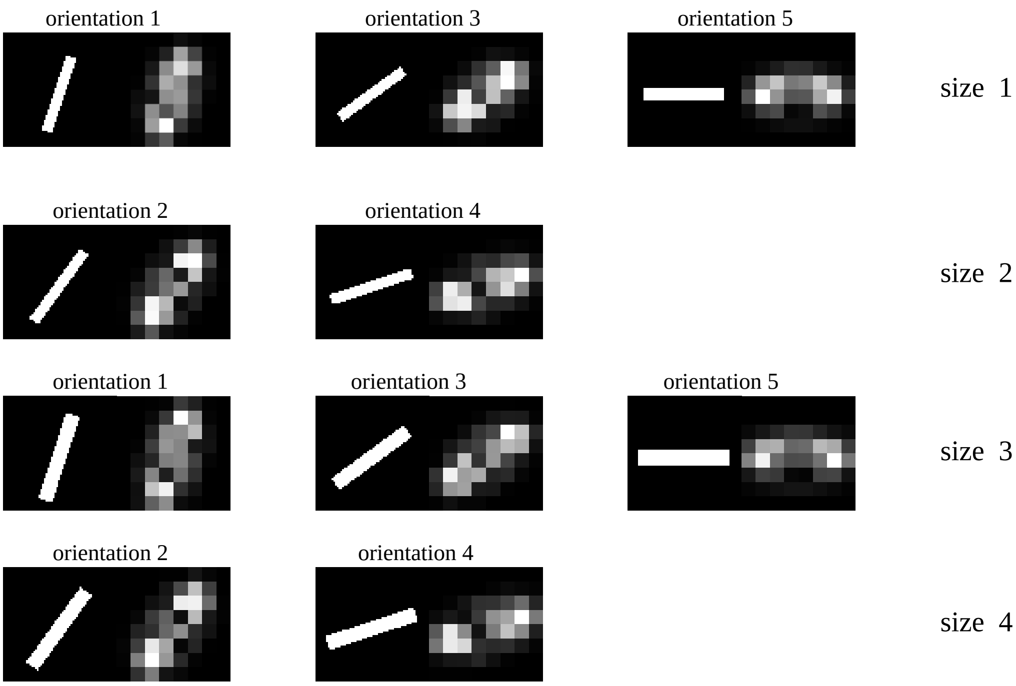

Figure 17: Bars - Training data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

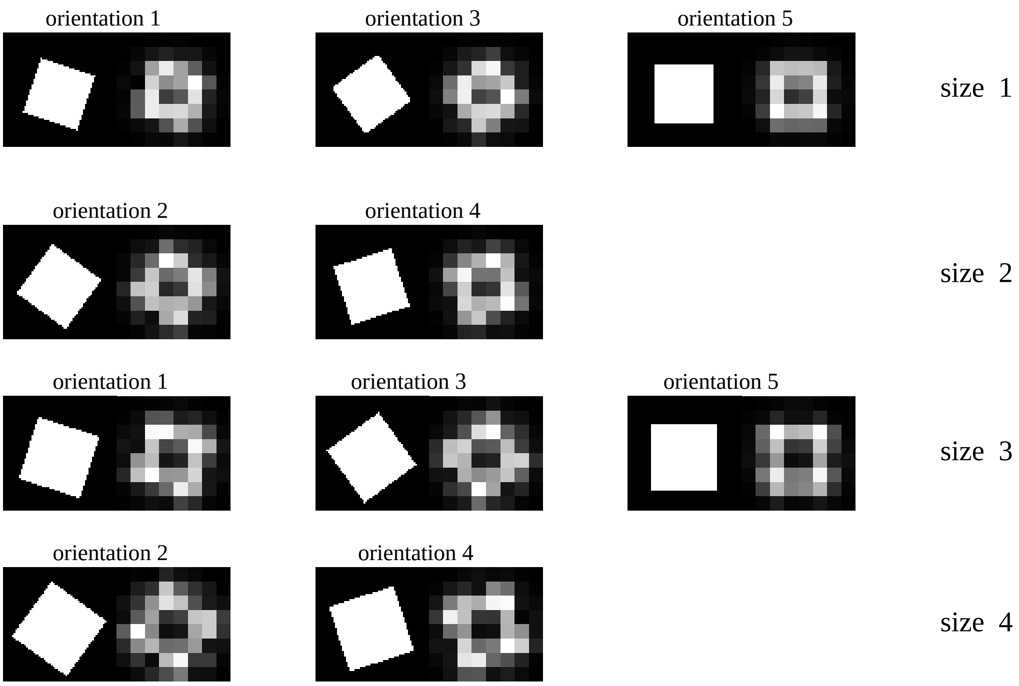

Figure 18: Squares - Training data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

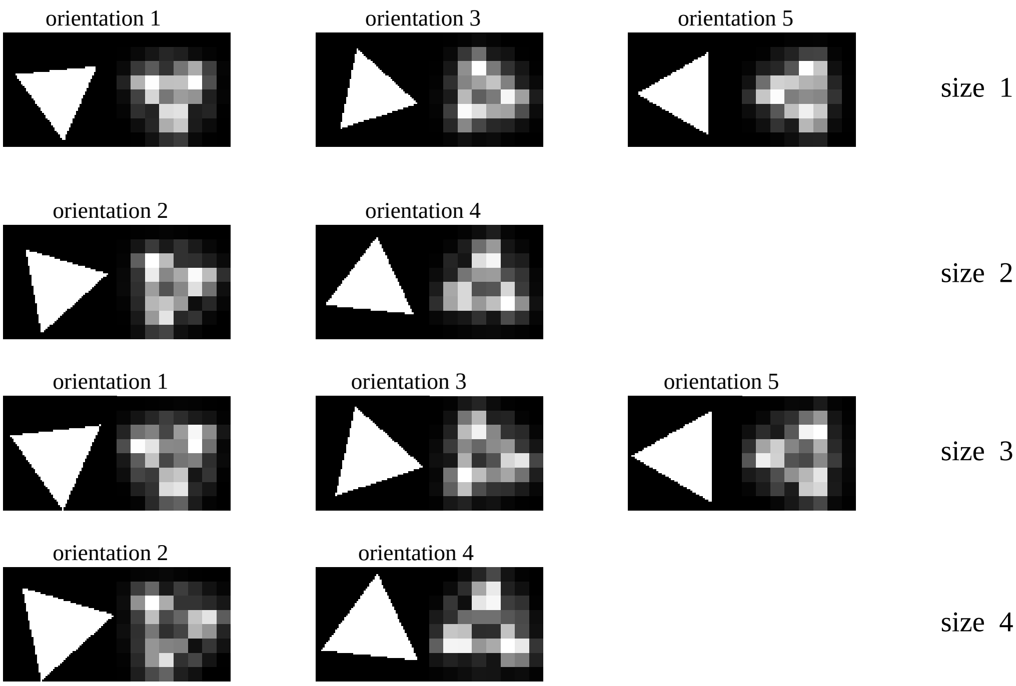

Figure 19: Triangles - Training data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

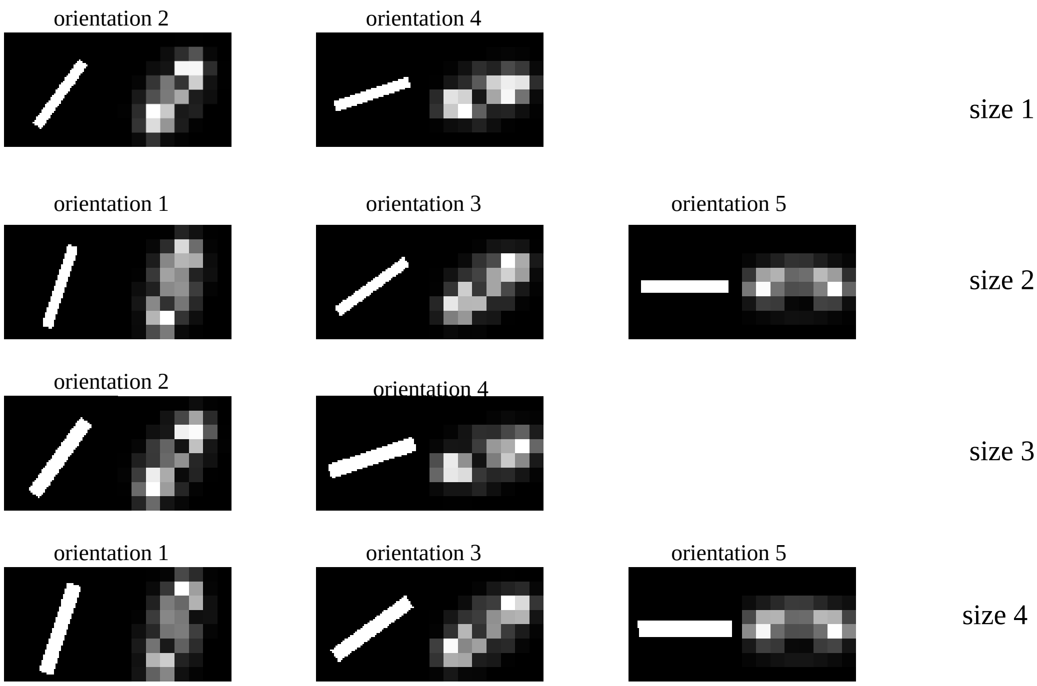

Figure 20: Bars - Testing data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

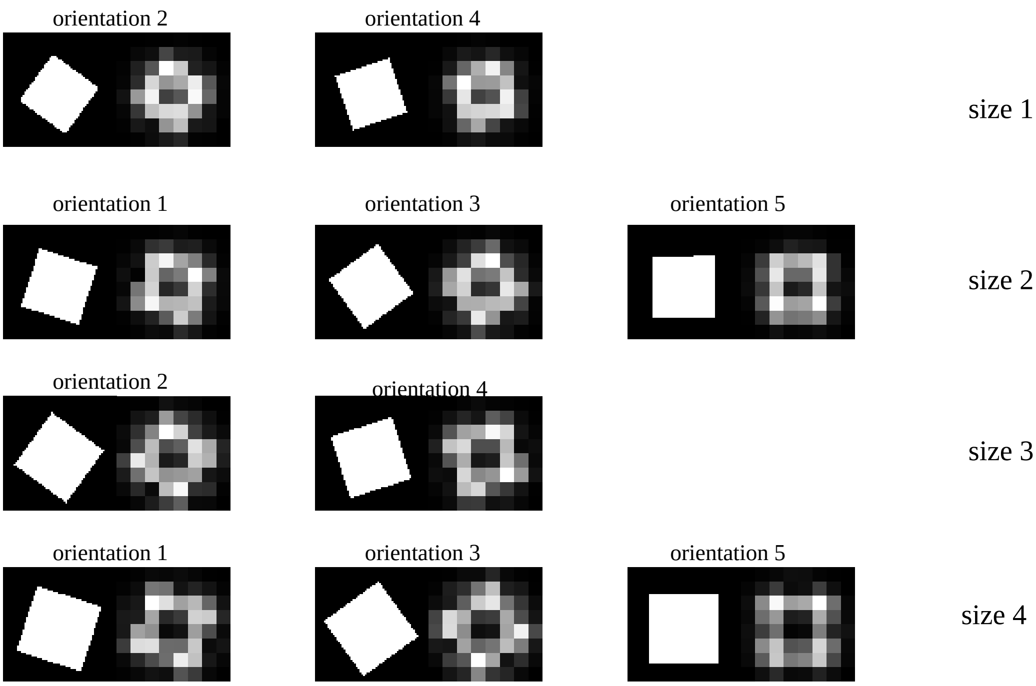

Figure 21: Squares - Testing data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

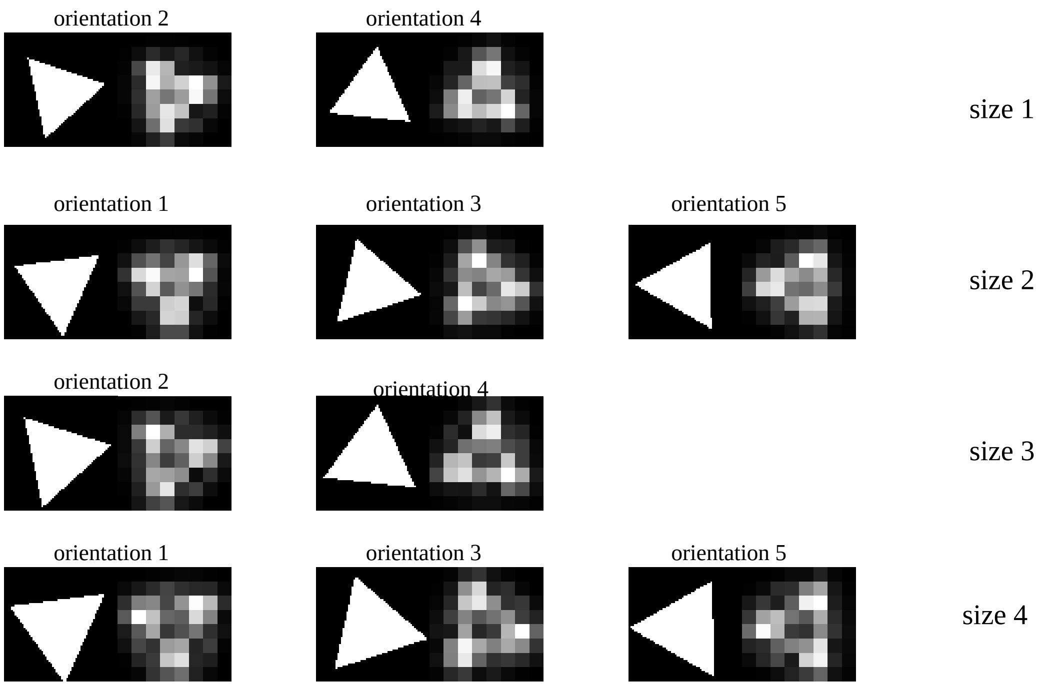

Figure 22: Triangles - Testing data. The original image is shown to the left of each image pair. The subsampled corner response is shown on the right.

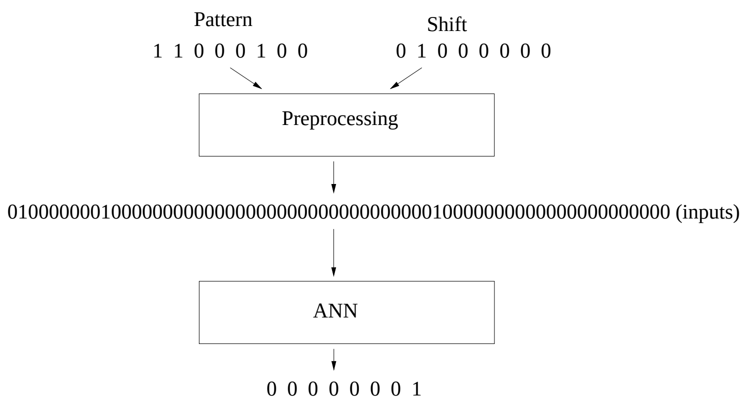

Figure 23: Example of pattern shifter data.

3.5 ANN for the control of pattern shifting

On a rather different note, a network that will perform a shift of a one dimensional pattern is developed. Eventually there will be a need for a two dimensional pattern shifter that attempts to align two patterns - a stored object representation with a part of an image in the visual field, for example.

Data is generated which comprises: a randomly generated binary pattern to represent the bits that are to be shifted and a randomly generated binary pattern that represents the number of places that the bits must shift to the right. These two patterns are combined to form a second order input to the network. Another pattern that represents the correctly shifted bit pattern was calculated for the target output pattern. For the network inputs, two patterns are generated randomly: the pattern to be shifted and the shift bit pattern - both patterns being 8 bits long. The pattern to be shifted consists of a randomly generated bit pattern of the numbers 0 to 255; numerically this gives all the permutations of an 8 bit pattern. The shift bit pattern is generated so that only one bit is ever set. An example is given in figure 23. The second-order input is created by calculating the product of each of the 64 pairs. The network is a feedforward ANN trained with BP, and with no hidden layers - see figure 24.

200 data sets were randomly generated for training. The total number of possible input patterns is $256 \times 8 = 2048$ - that is, 8 possible shifts of each of the 256 patterns. Thus, approximately 10% of the whole set were used for training the network. Convergence occurred after 135 epochs with a learning rate of 0.05, an inertia rate of 0.95 and and error bound of 0.05. Another (different) data file containing 200 data sets was generated for testing the network. With the error bound set at 0.1, only 2 data sets were incorrectly classified, and these two were also correctly classified when the error bound was raised to 0.15. This network was thus judged to be giving reliable generalisation to novel data. The network was trained with varying amounts of

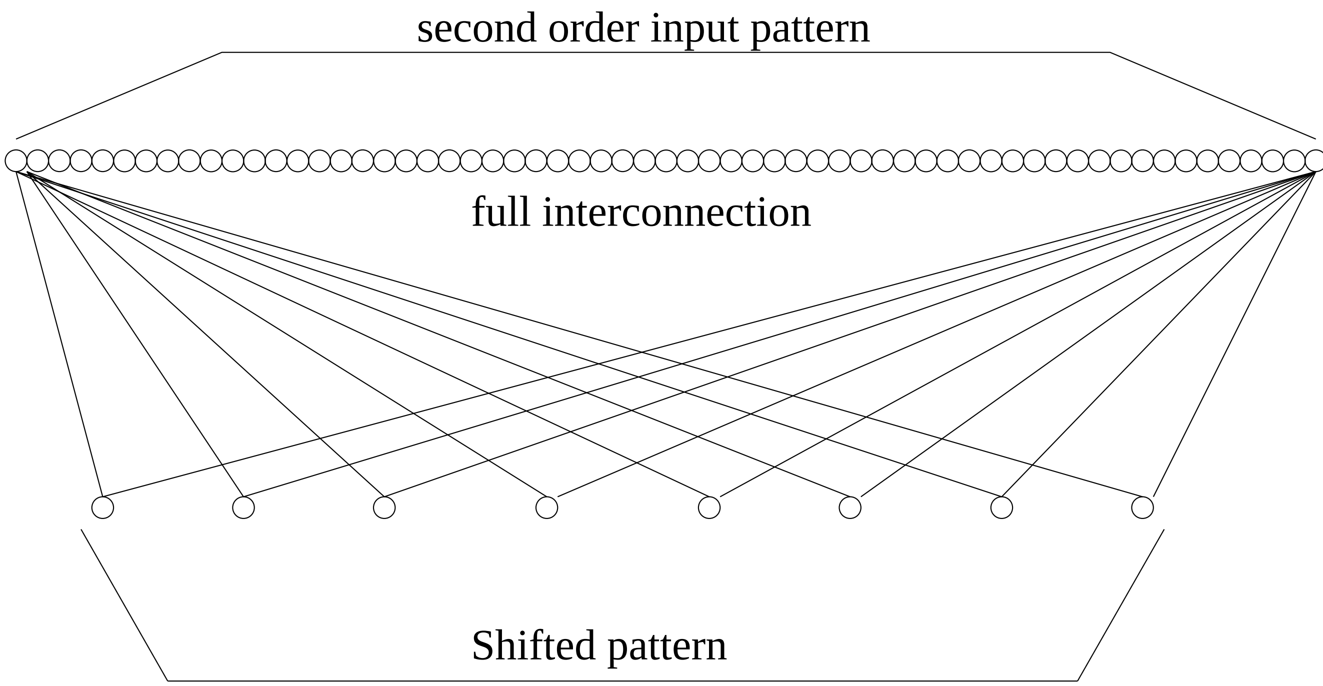

Figure 24: Architecture of the pattern shifter network.

uniformly distributed noise to test its robustness. The results are shown in figure 25. Networks trained with varying amounts of noise were tested with data that also had noise added. The results of this are shown in figure 26.

One of the questions that one might usefully ask about the noise-trained networks, is whether they are better equipped to handle the noisy testing data. The answer is clearly affirmative, and the network that was trained with the noisiest data performs best with the noisy testing data.

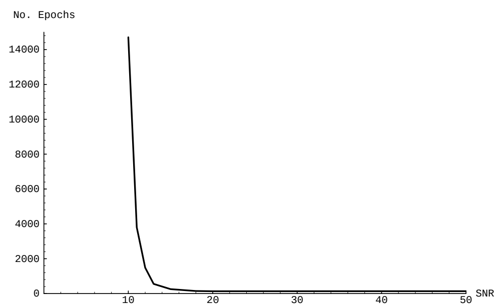

Figure 25: The effect of noisy data on network convergence during training.

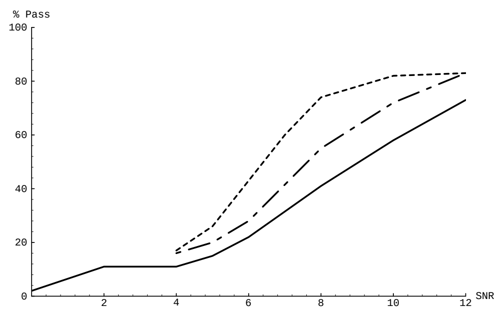

Figure 26: The effects of noisy data on testing for various networks.

4 Evaluation and Further Work

From the experimental work presented in the previous section, there are two pieces of work that might, with further development, be used in the future. One of these is the third-order rotation invariant network. This seems to perform the task for which it was designed though it has not yet been tested for robustness to noise or occlusion. This robustness is acclaimed for HONNs by other workers. The third-order network also needs to be trained and tested with a wider variety of shapes so that the response becomes more finely tuned. The third-order network also needs to be tried with some larger levels of the pyramid. For these larger levels, the number of sectors into which the image is split will need to be increased. It should be noted that the method for splitting images into sectors is most suited to simple shapes. Because the sectors are so much wider towards the edge of the image, any detail in these areas will be 'blurred'. This effect will be worst where the number of sectors is small in relation to the resolution of the image.

The second piece of work that might be further developed is that of the ANN that was developed for pattern shifting. The network needs to be able to shift a two dimensional pattern for use with image data. The Experimental Work section describes the difficulties that are associated with the large number of inputs generated by higher order networks. For images larger than 8×8 this is liable to cause difficulties. It may be necessary to use a coarse coding method to overcome this problem. Splitting the image into sectors would cause problems because the image becomes degraded using this method. In addition, the shifter network could be developed to perform rotational transformations. Scaling too may be neccessary. Although the plan is to represent objects at multiple resolutions, the incoming image of an object is unlikely to appear at precisely the right size for the pyramid. There will therefore be a need to normalise the size of incoming image objects.

Once the facilities for scaling, rotation and translation exist the development of the system will be towards coordinating the interaction between the database (what) part of the network and the attention system (where). It is not clear at this point precisely how this will be accomplished.

References

[1] A. H. Bhalerao. *Multiresolution Image Segmentation.* PhD thesis, Department of Computer Science, The University of Warwick, UK, October 1991.

[2] P. J. Burt and E. H. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communication*, COM-31:532–540, 1983.

[3] G. A. Carpenter and S. Grossberg. A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *CVGIP*, 37:54–115, 1987a.

[4] S. Clippingdale. *Multiresolution Image Modelling and Estimation.* PhD thesis, Department of Computer Science, The University of Warwick, UK, September 1988.

[5] P. S. Goldman-Rakic. Working Memory and the Mind. *Scientific American*, pages 73–79, September 1992.

[6] C. M. Gray, P. Konig, A. K. Engel, and W. Singer. Oscillatory Responses in Cat Visual Cortex Exhibit Inter-columnar Synchronisation Which Reflects Global Stimulus Properties. *Nature*, 338:334–, 1989.

[7] D. H. Hubel. *Eye, Brain, and Vision.* Scientific American Library, 1988.

[8] J. E. Hummel and I. Biederman. Dynamic Binding in a Neural Network for Shape Recognition. *Psychological Review*, 99:480–517, 1992.

[9] A. King. Multiresolution image analysis based on local symmetries. Technical Report RR248, University of Warwick, 1993.

[10] S. M. Kosslyn. Seeing and imagining in the cerebral hemispheres. *Psychological Review*, 94:148–175, 1987.

[11] D. Psaltis, A. Sideris, and A.A. Yamamura. A Multilayered Neural Network Controller. *IEEE Control Systems Magazine*, pages 17–21, April 1987.

[12] R. P. N. Rao and D. H. Ballard. An Active Vision Architecture based on Iconic Representations. Technical Report 548, University of Rochester, USA, 1995.

[13] C. R. Rosenberg and T. J. Sejnowski. Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1:145–168, 1987.

[14] W. A. C. Schmidt and J. P. Davis. Pattern Recognition Properties of Various Feature Spaces for Higher Order Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:795–801, 1993.

[15] J. P. Siebert and I. Eising. Scale-Space Recognition Based on the Retino-Cortical Transsform. In *IEEIPA*, 1995.

[16] L. Spirkovska and M. B. Reid. Robust Position, Scale, and Rotation Invariant Object Recognition using Higher Order Neural Networks. *Pattern Recognition*, 25:975–985, 1992.

[17] A. Treisman. Perceptual Grouping and Attention in Visual Search for Features and Objects. *Journal of Experimental Psychology*, 8:184–214, 1982.

[18] R. R. Yager. Modelling and Formulating Fuzzy Knowledge Bases Using Neural Networks. *Neural Networks*, 7:1273–1283, 1994.

[19] H. C. Yang. *Multiresolution Neural Networks for Image Edge Detection and Restoration.* PhD thesis, Department of Computer Science, The University of Warwick, UK, 1994.